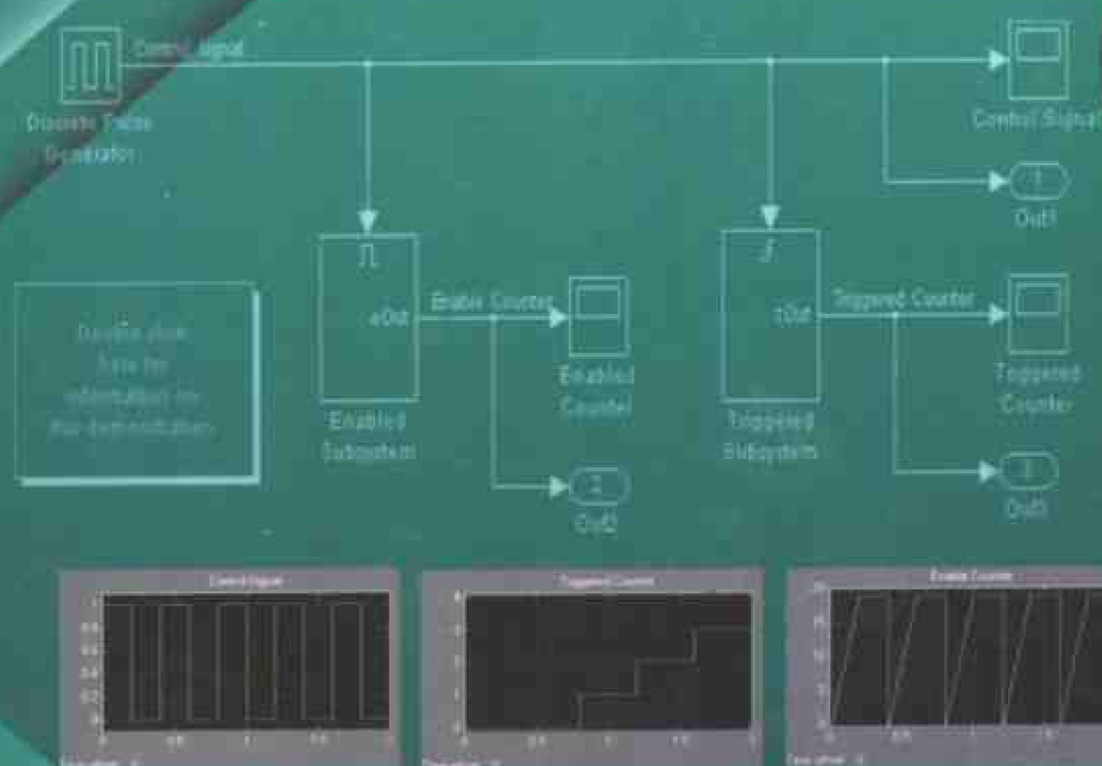
21世纪工程应用计算机技术丛书

基于MATLAB的动态模型与系统仿真工具

——Simulink 3.0/4.X



邱晓林　李天柁
弟宇鸣　肖　刚　编著

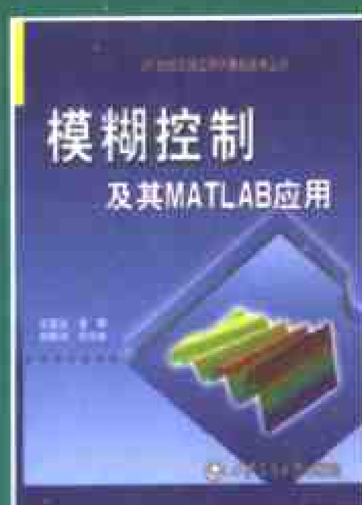


西安交通大学出版社

基于MATLAB的动态模型与系统仿真工具

——Simulink 3.0/4.X

21世纪工程应用计算机技术丛书



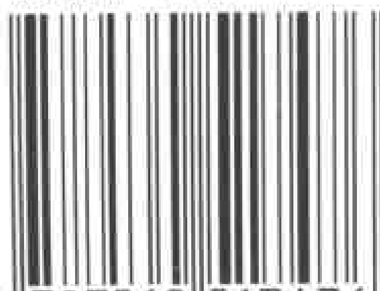
模糊控制及其MATLAB应用
张国良 曾 静 柯熙政 邓方林 编著
2002年11月出版
15.00元



基于MATLAB的动态模型与系统仿真工具——Simulink 3.0/4.X
邱晓林 李天柁 弟宇鸣 肖 刚 编著
2003年10月出版
30.00元

总结工程应用成果　提供丰富实例分析
整理相关文献资料　启发新的探索思路

□责任编辑／贺峰涛 [illegible]
□文字编辑／张立文
□装帧设计／阎 [illegible]
□版式设计／程文卫

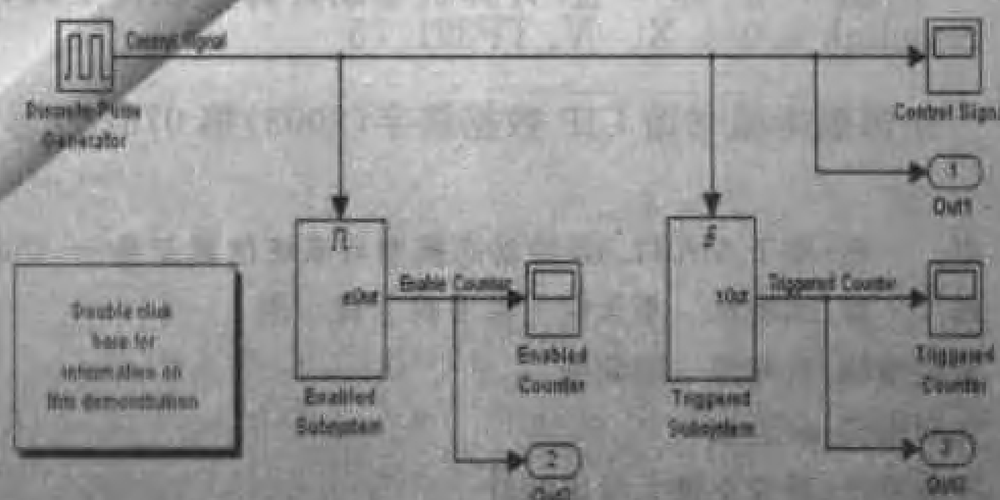




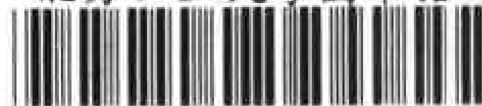
ISBN 7-5605-1747-1/TP·353　定价：30.00元

基于MATLAB的动态模型与系统仿真工具

——Simulink 3.0/4.X



邱晓林　李天柁
弟宇鸣　肖　刚　编著





西安交通大学出版社
·西　安·

内 容 简 介

本书以 Simulink 3.0 为基础，全面、系统地介绍了 Simulink 系统和模型的基本概念，仿真模型的创建、调试和优化方法以及各种相关信息，并介绍了 Simulink 4.X 的新增和改进功能。主要内容包括：Simulink 入门必读、系统操作命令、仿真模型编辑器、仿真模型动态调试器、仿真运行与结果分析、高级仿真模型创建方法、库模块以及 Simulink 的最新发展。附录给出了 Simulink 的模型、模块参数及模型文件格式。

本书具有内容丰富、涵盖面广、适用多种平台和举例多的特点，既适合于 MATLAB 的初学者，也适用于各种仿真高手，可作为科研技术人员、高校师生和工程技术人员应用 MATLAB 解决实际问题的参考书。

图书在版编目(CIP)数据

基于 MATLAB 的动态模型与系统仿真工具——Simulink 3.0/4.X/邱晓林等编著．—西安：西安交通大学出版社，2003.10
(21 世纪工程应用计算机技术丛书)
ISBN 7－5605－1747－1

Ⅰ．基… Ⅱ．邱… Ⅲ．计算机辅助计算－软件工具，Simulink 3.0/4.X Ⅳ．TP391.75

中国版本图书馆 CIP 数据核字(2003)第 076280 号

书　　名：基于 MATLAB 的动态模型与系统仿真工具——Simulink 3.0/4.X
编　　著：邱晓林　李天枨　弟宇鸣　肖　刚
策划编辑：贺峰涛　屈晓燕
文字编辑：宗立文
出版发行：西安交通大学出版社
地　　址：西安市兴庆南路 25 号(邮编：710049)
网　　址：http://unit.xjtu.edu.cn/unit/jtupress
电　　话：(029)2668357　2667874(发行部)
　　　　　(029)2668315　2669096(总编部)
电子信箱：eibooks@163.com
印　　刷：陕西宝石兰印务有限责任公司
版　　次：2003 年 10 月第 1 版　2003 年 10 月第 1 次印刷
开　　本：787mm×1092mm　1/16
印　　张：21.5
印　　数：0 001～3 000
字　　数：512 千字
书　　号：ISBN 7－5605－1747－1/TP・353
定　　价：30.00 元

前　言

MATLAB 的初学者，尤其是工程设计人员，可能有这样的体会：虽然使用 MATLAB 语言（命令）能较为方便地进行各种复杂的数学运算，但系统模型的建立、仿真以及程序的调试仍然是一件颇花费时间的事情。

Simulink 是 MATLAB 提供的一个用于对动态系统进行建模和仿真的软件包，具有丰富和灵活的功能。有了它，用户就可以将自己的计算机变成一个方便快捷的、面向各种系统的建模和分析实验室，从而解决相应的问题。

假如您是一名电气工程设计师，一定会对在电路设计过程中屡次推翻原方案深感无奈。当您使用 Simulink 后就会发现，无论对哪种系统（连续、离散或混合系统，甚至是多制采样率系统），您都只需在 Simulink 模块库中找到相应的功能模块，选中它们，然后将它们用线段连接起来，就可以构建任何一种由功能模块和连线组成的电子学系统的模型。通过改变模块和连线的参数并借助于 Simulink 调试器，您可以随心所欲地对模型进行修改、调试和仿真分析，直到满意为止。这样不仅能减少方案设计的次数，提高效率，降低设计成本，而且整个过程就像是搭积木一样方便有趣。

我们根据自己使用 Simulink 的经验，编写了本书，试图向读者全面、系统地介绍 Simulink 的使用方法。本书以 Simulink 3.0 为基础，主要介绍了 Simulink 系统和模型的基本概念，仿真模型的创建、调试和优化方法以及各种相关信息，并介绍了 Simulink 4.X 的新增和改进功能。本书内容包括：

第 1 章　Simulink 入门必读。通过运行一个示例模型，介绍 Simulink 的基础知识，帮助读者迅速了解使用 Simulink 以及建立模型的基本步骤。

第 2 章　Simulink 系统操作命令。介绍从 MATLAB 命令窗口或利用 M 文件创建和修改模型的命令及其使用方法，并列出了所有关于模型命令的信息。

第 3 章　Simulink 仿真模型编辑器。详细介绍 Simulink 仿真模型的基本编辑功能，并通过示例说明利用 GUI 编辑仿真模型的方法。

第 4 章　Simulink 仿真模型动态调试器。介绍使用 Simulink 调试器调试 Simulink 模型的方法，并给出了调试命令的应用实例。

第 5 章　Simulink 仿真运行与结果分析。介绍仿真的运行模式，仿真器及其选择建议，线性、非线性、离散及混合系统模型的仿真及相关函数。

第 6 章　Simulink 高级仿真模型创建方法。介绍利用 Simulink 封装编辑器创建用户模块，采用封装技术以定制封装模块的方法。同时介绍了一些 Simulink 和 MATLAB 中用于仿真的过程观察和结果分析的特性。

第 7 章　Simulink 库模块。给出了所有库模块的功能及参数特性供查阅、选择。

第 8 章　Simulink 的最新发展。主要介绍了 Simulink 4. X 版本与 Simulink 3. 0 的不同之处，为读者了解、学习和使用更新 Simulink 版本以解决特殊仿真问题提供参考。

附录　模型、模块参数及模型文件格式。给出了模块、模型的共有参数和特有参数，并通过一个示例介绍了 Simulink 仿真模型的组成，给出了 M 文件源代码。

本书具有以下几个特点：

1. 面向所有读者。如果您是一名初学者，只要仔细阅读本书并充分发挥自身的才能，就能成为一名仿真高手；如果您是一位仿真高手，本书的许多内容可以帮助您更上一层楼。

2. 内容丰富，涵盖面广。本书以目前主流产品 Simulink 3. 0 的内容为主构成，并增加了由 MathWorks 公司最新发布的 Simulink 4. X(随 MATLAB 6. X 发布)的新增及改进功能，是一本全面介绍 Simulink 的参考书。

3. 适用多种平台。虽然目前常用的操作系统是 Windows 系统，但仍然有一些用户使用的是 UNIX 等系统平台，本书考虑到了这一点，随时指出它们的不同。

4. 举例多、实用性强。本书提供了众多的示例，帮助读者更方便地学习。

我们希望本书能引导您进入 Simulink 的大门，使您畅游在仿真世界中，领略无限风光。

感谢王军虎对书中所涉及的应用程序进行了验证。

西安交通大学自控所的韩九强教授审阅了全部书稿并提出了许多宝贵的意见和建议，在此表示衷心地感谢。

本书的出版得到了第二炮兵工程学院训练部的资助和西安交通大学出版社的帮助，在此一并表示诚挚的谢意。

由于编者水平所限，书中错误在所难免，敬请读者不吝赐教。

编　者

2003 年 2 月

目 录

第 1 章 Simulink 入门必读 ……(1)

1.1 Simulink 的特点 ……(1)

1.2 应用程序工具箱介绍 ……(2)

1.3 Simulink 实时工作间 ……(4)

1.3.1 用途 ……(4)

1.3.2 主要特性 ……(5)

1.3.3 运行环境 ……(5)

1.4 Ada 扩展的实时工作间 ……(5)

1.5 模块集 ……(6)

1.5.1 DSP 模块集 ……(6)

1.5.2 定点模块集 ……(6)

1.5.3 非线性控制设计模块集 ……(7)

1.5.4 电力系统模块集 ……(7)

1.6 模型演示 ……(7)

1.6.1 运行演示模型 ……(7)

1.6.2 演示模型描述 ……(8)

1.6.3 初试身手 ……(9)

1.6.4 关于本演示程序的进一步探讨 ……(9)

1.6.5 其它演示程序 ……(10)

1.7 创建一个简单模型 ……(10)

第 2 章 Simulink 系统操作命令

2.1 命令及要求概述 ……(17)

2.1.1 命令及其功能 ……(17)

2.1.2 要求一:指定命令的参数 ……(18)

2.1.3 要求二:指定执行对象的路径 ……(18)

2.1.4 有关说明 ……(18)

2.2 用命令建模实例 ……(18)

2.3 add_block 命令 ……(20)

2.4 add_line 命令 ……(21)

2.5 bdclose 命令 ……(21)

2.6 bdroot 命令 ……(22)

2.7 close_system 命令 …… (22)
2.8 delete_block 命令 …… (23)
2.9 delete_line 模块 …… (24)
2.10 find_system 命令 …… (24)
2.11 gcb 命令 …… (26)
2.12 gcbh 命令 …… (27)
2.13 gcs 命令 …… (27)
2.14 get_param 命令 …… (28)
2.15 new_system 命令 …… (29)
2.16 open_system 命令 …… (30)
2.17 replace_block 命令 …… (30)
2.18 save_system 命令 …… (31)
2.19 set_param 命令 …… (32)
2.20 simulink 命令 …… (32)
2.21 simulink3 命令 …… (33)

第 3 章 Simulink 仿真模型编辑器

3.1 Simulink 模型编辑概述 …… (34)
3.1.1 进入 Simulink …… (34)
3.1.2 构建新模型 …… (35)
3.1.3 编辑已有的模型 …… (35)
3.1.4 Simulink 命令的输入 …… (35)
3.1.5 Simulink 窗口 …… (36)
3.1.6 状态栏 …… (36)
3.1.7 放大和缩小模型图 …… (37)
3.2 选择对象 …… (37)
3.2.1 选择一个对象 …… (37)
3.2.2 选择多个对象 …… (37)
3.3 模块及其编辑 …… (38)
3.3.1 模块信息提示 …… (38)
3.3.2 虚拟模块 …… (38)
3.3.3 模块的复制和移动 …… (39)
3.3.4 设定模块参数 …… (40)
3.3.5 模块的属性对话框 …… (41)
3.3.6 删除模块 …… (42)
3.3.7 改变模块的方向 …… (42)
3.3.8 调整模块大小 …… (43)
3.3.9 编辑模块名 …… (43)
3.3.10 在模型图标下显示模块参数 …… (44)
3.3.11 断开模块 …… (44)

3.3.12 矢量输入和输出 …… (44)
3.3.13 输入和参数的标量扩展 …… (44)
3.3.14 指派模块优先级 …… (45)
3.3.15 使用阴影 …… (45)
3.4 库的概念及操作 …… (45)
3.4.1 术语 …… (46)
3.4.2 创建库 …… (46)
3.4.3 修改库 …… (46)
3.4.4 向模型中复制库模块 …… (46)
3.4.5 更新已链接的模块 …… (47)
3.4.6 断开与库模块的链接 …… (47)
3.4.7 为参考模块寻找库模块 …… (48)
3.4.8 获得库模块信息 …… (48)
3.4.9 浏览模块库 …… (48)
3.5 线 …… (49)
3.5.1 在两个模块间画线 …… (49)
3.5.2 画分支线 …… (49)
3.5.3 画线段 …… (50)
3.5.4 显示线的宽度 …… (51)
3.5.5 在线上插入模块 …… (52)
3.5.6 信号标签及其编辑和传递特性 …… (52)
3.5.7 信号属性及其设置 …… (53)
3.6 注释 …… (54)
3.6.1 创建模型注释 …… (54)
3.6.2 移动注释 …… (55)
3.6.3 编辑注释 …… (55)
3.6.4 删除注释 …… (55)
3.6.5 更改注释的字体 …… (55)
3.7 数据类型 …… (55)
3.7.1 Simulink 支持的数据类型 …… (56)
3.7.2 支持数据和数字信号类型的模块 …… (56)
3.7.3 指定模块参数的数据类型 …… (58)
3.7.4 创建一个指定数据类型的信号 …… (59)
3.7.5 显示端口数据类型 …… (59)
3.7.6 数据类型传递 …… (59)
3.7.7 数据类型规则 …… (59)
3.7.8 激活严格布尔类型检测 …… (60)
3.7.9 转换信号的数据类型 …… (60)
3.7.10 转换参数的数据类型 …… (60)
3.8 复信号工作方式 …… (61)

3.9 鼠标和键盘操作简介 ………… (61)
3.10 创建子系统 ………… (63)
3.10.1 通过添加 Subsystem 模块创建子系统 ………… (63)
3.10.2 通过组合已有的模块创建子系统 ………… (63)
3.10.3 子系统端口标签 ………… (64)
3.10.4 使用调回例程 ………… (64)
3.11 有关模型构建的提示 ………… (66)
3.12 构建方程式模型 ………… (66)
3.12.1 摄氏-华氏温度转换模型 ………… (66)
3.12.2 建立一个简单的连续系统模型 ………… (67)
3.13 保存模型 ………… (69)
3.14 打印模型图 ………… (69)
3.14.1 打印对话框 ………… (69)
3.14.2 使用命令打印 ………… (70)
3.14.3 指定纸张大小和打印方向 ………… (71)
3.14.4 定位和调整图表大小 ………… (71)
3.15 模型浏览器 ………… (72)
3.15.1 Windows 系统的模型浏览器 ………… (72)
3.15.2 使用 UNIX 系统模型浏览器 ………… (72)
3.16 追踪模型版本 ………… (74)
3.16.1 指定当前用户 ………… (74)
3.16.2 模型属性对话框 ………… (75)
3.16.3 创建模型的变化历史记录 ………… (78)
3.16.4 版本控制属性 ………… (79)
3.17 退出 Simulink ………… (80)

第 4 章 Simulink 仿真模型动态调试器

4.1 调试器使用概述 ………… (81)
4.1.1 启动调试器 ………… (81)
4.1.2 获得帮助 ………… (82)
4.1.3 键入命令 ………… (82)
4.1.4 模块指数 ………… (82)
4.1.5 访问 MATLAB 工作空间 ………… (82)
4.2 步进运行仿真 ………… (83)
4.2.1 模块步进 ………… (83)
4.2.2 时间步步进 ………… (84)
4.2.3 断点步进 ………… (84)
4.2.4 不间断运行仿真 ………… (84)
4.3 设置断点 ………… (84)
4.3.1 模块断点 ………… (85)

4.3.2 设置时间步断点……………………………………………………（86）
4.3.3 对非限定值设置断点……………………………………………………（86）
4.3.4 设置限定步长步进的断点……………………………………………………（86）
4.3.5 设置过零断点……………………………………………………（86）
4.4 显示仿真信息……………………………………………………（87）
4.4.1 显示模块 I/O ……………………………………………………（87）
4.4.2 显示代数环信息……………………………………………………（88）
4.4.3 显示系统状态……………………………………………………（88）
4.4.4 显示集成信息……………………………………………………（89）
4.5 显示模型信息……………………………………………………（89）
4.5.1 显示模型的模块执行顺序……………………………………………………（89）
4.5.2 显示一个模块……………………………………………………（89）
4.5.3 显示模型的非虚拟系统……………………………………………………（89）
4.5.4 显示模型的非虚拟模块……………………………………………………（90）
4.5.5 显示含有潜在过零模块……………………………………………………（90）
4.5.6 显示代数环……………………………………………………（91）
4.5.7 显示调试工具的设置……………………………………………………（91）
4.6 调试器命令总汇……………………………………………………（91）
4.6.1 ashow 命令……………………………………………………（93）
4.6.2 atrace 命令 ……………………………………………………（93）
4.6.3 bafter 命令 ……………………………………………………（93）
4.6.4 break 命令 ……………………………………………………（93）
4.6.5 bshow 命令 ……………………………………………………（94）
4.6.6 clear 命令 ……………………………………………………（94）
4.6.7 continue 命令 ……………………………………………………（94）
4.6.8 disp 命令……………………………………………………（94）
4.6.9 help 命令 ……………………………………………………（95）
4.6.10 ishow 命令 ……………………………………………………（95）
4.6.11 minor 命令 ……………………………………………………（95）
4.6.12 nanbreak 命令 ……………………………………………………（95）
4.6.13 next 命令 ……………………………………………………（95）
4.6.14 probe 命令 ……………………………………………………（95）
4.6.15 quit 命令 ……………………………………………………（96）
4.6.16 run 命令 ……………………………………………………（96）
4.6.17 slist 命令 ……………………………………………………（96）
4.6.18 states 命令 ……………………………………………………（96）
4.6.19 systems 命令 ……………………………………………………（96）
4.6.20 status 命令 ……………………………………………………（97）
4.6.21 step 命令 ……………………………………………………（97）
4.6.22 stop 命令 ……………………………………………………（97）

4.6.23 tbreak 命令 …… (97)
4.6.24 trace 命令 …… (97)
4.6.25 undisp 命令 …… (97)
4.6.26 untrace 命令 …… (98)
4.6.27 xbreak 命令 …… (98)
4.6.28 zcbreak 命令 …… (98)
4.6.29 zclist 命令 …… (98)

第 5 章 Simulink 仿真运行与结果分析

5.1 仿真的运行方式比较 …… (99)
5.1.1 使用菜单命令 …… (99)
5.1.2 从命令行运行仿真 …… (100)
5.2 使用菜单命令运行仿真 …… (100)
5.2.1 设置仿真参数和选择仿真器(Solver) …… (100)
5.2.2 应用仿真参数 …… (100)
5.2.3 运行仿真 …… (100)
5.2.4 仿真诊断(Simulation Diagnostics)对话框 …… (101)
5.3 仿真参数对话框 …… (102)
5.3.1 Solver 选项及其设置 …… (102)
5.3.2 Workspace I/O 选项设置 …… (107)
5.3.3 Diagnostics(诊断)选项设置 …… (111)
5.4 提高仿真性能和精度 …… (113)
5.4.1 提高仿真速度 …… (113)
5.4.2 提高仿真精度 …… (114)
5.5 在命令行输入命令运行仿真 …… (114)
5.5.1 使用 sim 命令 …… (114)
5.5.2 使用 set_param 命令 …… (114)
5.5.3 sim 命令 …… (115)
5.5.4 simset 命令 …… (116)
5.5.5 simget 命令 …… (118)
5.6 仿真结果的分析 …… (119)
5.6.1 使用 Scope 模块观察输出信号 …… (119)
5.6.2 使用返回变量 …… (119)
5.6.3 使用 To Workspace 模块 …… (119)
5.7 线性化与线性分析 …… (120)
5.7.1 线性模型 …… (120)
5.7.2 非线性模型 …… (121)
5.7.3 离散系统或者混合连续离散系统 …… (121)
5.8 平衡点的确定 …… (122)
5.9 linfun 函数 …… (123)

5.9.1 命令用途 ······ (123)
5.9.2 命令格式 ······ (123)
5.9.3 命令参数 ······ (123)
5.9.4 命令描述 ······ (124)
5.10 trim 函数 ······ (126)
5.10.1 命令用途 ······ (126)
5.10.2 命令格式 ······ (126)
5.10.3 命令描述 ······ (126)
5.10.4 应用举例 ······ (127)
5.10.5 限制条件 ······ (129)
5.10.6 命令算法 ······ (129)

第 6 章 Simulink 高级仿真模型创建方法

6.1 关于模块定制和子系统的封装技术 ······ (130)
6.2 一个封装子系统的示例 ······ (130)
6.2.1 创建封装对话框中的提示及相关信息 ······ (132)
6.2.2 创建模块描述和帮助文本 ······ (133)
6.2.3 创建模块图标 ······ (133)
6.2.4 封装方法小结 ······ (134)
6.3 封装编辑器(Mask Editor) ······ (134)
6.3.1 初始化(Initialization)选项 ······ (135)
6.3.2 Icon 选项 ······ (140)
6.3.3 Documentation 选项 ······ (146)
6.3.4 为封装模块创建动态对话 ······ (147)
6.4 条件执行子系统 ······ (148)
6.4.1 使能子系统 ······ (149)
6.4.2 触发子系统 ······ (151)
6.4.3 触发子系统可以包含的模块 ······ (153)
6.5 触发加使能子系统 ······ (153)
6.6 Simulink 仿真原理 ······ (155)
6.6.1 Simulink 工作程序 ······ (155)
6.6.2 离散时间系统注意事项 ······ (161)

第 7 章 Simulink 库模块

7.1 Simulink 模块库 ······ (165)
7.2 库模块预览 ······ (165)
7.3 库模块相关说明 ······ (169)
7.4 输入源库模块 ······ (170)
7.4.1 Band-Limited White Noise 模块 ······ (170)
7.4.2 Chirp Signal 模块 ······ (171)

7.4.3 Clock 模块 …… (172)
7.4.4 Constant 模块 …… (173)
7.4.5 Digital Clock 模块 …… (173)
7.4.6 Discrete Pulse Generator 模块 …… (174)
7.4.7 From File 模块 …… (175)
7.4.8 From Workspace 模块 …… (176)
7.4.9 Pulse Generator 模块 …… (178)
7.4.10 Ramp 模块 …… (179)
7.4.11 Random Number 模块 …… (179)
7.4.12 Repeating Sequence 模块 …… (180)
7.4.13 Signal Generator 模块 …… (181)
7.4.14 Sine Wave 模块 …… (182)
7.4.15 Step 模块 …… (184)
7.4.16 Uniform Random Number 模块 …… (185)
7.5 接收器库模块 …… (186)
7.5.1 Display 模块 …… (186)
7.5.2 Scope 模块 …… (187)
7.5.3 Stop Simulation 模块 …… (193)
7.5.4 To File 模块 …… (194)
7.5.5 To Workspace 模块 …… (195)
7.5.6 XY Graph 模块 …… (197)
7.6 离散系统库模块 …… (198)
7.6.1 Discrete Filter 模块 …… (198)
7.6.2 Discrete State-Space 模块 …… (199)
7.6.3 Discrete-Time Integrator 模块 …… (200)
7.6.4 Discrete Transfer Fcn 模块 …… (203)
7.6.5 Discrete Zero-Pole 模块 …… (204)
7.6.6 First-Order Hold 模块 …… (205)
7.6.7 Unit Delay 模块 …… (206)
7.6.8 Zero-Order Hold 模块 …… (207)
7.7 连续系统库模块 …… (208)
7.7.1 Derivative 模块 …… (208)
7.7.2 Integrator 模块 …… (209)
7.7.3 Memory 模块 …… (211)
7.7.4 State-Space 模块 …… (212)
7.7.5 Transfer Fcn 模块 …… (213)
7.7.6 Transport Delay 模块 …… (215)
7.7.7 Variable Transport Delay 模块 …… (216)
7.7.8 Zero-Pole 模块 …… (217)
7.8 数学运算库模块 …… (218)

7.8.1 Abs 模块 …… (218)
7.8.2 Algebraic Constraint 模块 …… (219)
7.8.3 Combinatorial Logic 模块 …… (220)
7.8.4 Complex to Magnitude-Angle 模块 …… (222)
7.8.5 Complex to Real-Imag 模块 …… (222)
7.8.6 Dot Product 模块 …… (223)
7.8.7 Gain 模块 …… (224)
7.8.8 Logical Operator 模块 …… (225)
7.8.9 Magnitude-Angle to Complex 模块 …… (226)
7.8.10 Math Function 模块 …… (227)
7.8.11 Matrix Gain 模块 …… (228)
7.8.12 MinMax 模块 …… (229)
7.8.13 Product 模块 …… (230)
7.8.14 Real-Imag to Complex 模块 …… (231)
7.8.15 Relational Operator 模块 …… (232)
7.8.16 Rounding Function 模块 …… (233)
7.8.17 Sign 模块 …… (234)
7.8.18 Slider Gain 模块 …… (234)
7.8.19 Sum 模块 …… (235)
7.8.20 Trigonometric Function 模块 …… (236)
7.9 常用函数和查表库模块 …… (237)
7.9.1 Fcn 模块 …… (237)
7.9.2 Look-Up Table 模块 …… (239)
7.9.3 Look-Up Table (2－D)模块 …… (240)
7.9.4 MATLAB Fcn 模块 …… (242)
7.9.5 S-function 模块 …… (243)
7.10 非线性系统库模块 …… (244)
7.10.1 Backlash 模块 …… (244)
7.10.2 Coulomb and Viscous Friction 模块 …… (246)
7.10.3 Dead Zone 模块 …… (247)
7.10.4 Manual Switch 模块 …… (249)
7.10.5 Multiport Switch 模块 …… (249)
7.10.6 Quantizer 模块 …… (250)
7.10.7 Rate Limiter 模块 …… (251)
7.10.8 Relay 模块 …… (252)
7.10.9 Saturation 模块 …… (253)
7.10.10 Switch 模块 …… (254)
7.11 信号与系统库模块 …… (255)
7.11.1 Bus Selector 模块 …… (255)
7.11.2 Configurable Subsystem 模块 …… (256)

7.11.3 Data Store Memory 模块 …… (258)
7.11.4 Data Store Read 模块 …… (258)
7.11.5 Data Store Write 模块 …… (259)
7.11.6 Data Type Conversion 模块 …… (260)
7.11.7 Demux 模块 …… (261)
7.11.8 Enable 模块 …… (263)
7.11.9 From 模块 …… (264)
7.11.10 Function-Call Generator 模块 …… (265)
7.11.11 Goto 模块 …… (266)
7.11.12 Goto Tag Visibility 模块 …… (267)
7.11.13 Ground 模块 …… (268)
7.11.14 Hit Crossing 模块 …… (269)
7.11.15 IC 模块 …… (270)
7.11.16 Inport 模块 …… (271)
7.11.17 Merge 模块 …… (273)
7.11.18 Model Info 模块 …… (274)
7.11.19 Mux 模块 …… (276)
7.11.20 Outport 模块 …… (277)
7.11.21 Probe 模块 …… (279)
7.11.22 Selector 模块 …… (280)
7.11.23 Subsystem 模块 …… (281)
7.11.24 Terminator 模块 …… (282)
7.11.25 Trigger 模块 …… (282)
7.11.26 Width 模块 …… (283)

第 8 章 Simulink 的最新发展

8.1 进入 Simulink 4.X …… (284)
8.2 新增功能及模块 …… (284)
8.2.1 Simulink 编辑器 …… (286)
8.2.2 建模改进 …… (288)
8.2.3 Simulink 调试器 …… (289)
8.2.4 模块库 …… (292)
8.2.5 SB2SL …… (298)
8.3 Simulink 4.X 的运行工具 …… (298)
8.3.1 Simulink 4.0 的运行工具简介 …… (298)
8.3.2 Simulink 加速器 …… (298)
8.3.3 模型差异工具 …… (299)
8.3.4 运行档案器 …… (301)
8.3.5 模型覆盖率工具 …… (301)

附录 模型、模块参数及模型文件格式

A. 模型参数 ……………………………………………… (302)
B. 模块的共有参数 ……………………………………… (304)
C. 模块的特有参数 ……………………………………… (306)
D. 封装参数 ……………………………………………… (316)
E. 模型文件格式及示例 ………………………………… (317)

参考文献

第1章 Simulink 入门必读

Simulink 是 MATLAB 提供的主要工具之一，也是目前在动态系统的建模和仿真等方面应用最广泛的工具之一。全世界有成千上万的工程师都使用它建立动态系统模型，从而解决实际问题。通过对本工具的了解、熟悉、掌握及应用，相信读者一定会在自己的专业中大有作为。如果您是初次接触 Simulink，最好从本章开始阅读。

主要内容：介绍有关 Simulink 的基础知识和简单系统的建模实例。

学习目的：了解 Simulink 系统的基本概念，并对 Simulink 仿真有初步的认识。

1.1 Simulink 的特点

Simulink 是一个用来进行动态系统建模、仿真和分析的集成软件包。它不仅可以进行线性系统仿真，也可进行非线性系统仿真，既可以实现连续时间系统仿真，也可实现离散时间系统甚至混合连续-离散时间系统的仿真，它还支持多制采样率的系统仿真。

1. 建模方便、快捷

根据我们以前的经历，建模就意味着编程，例如要建立一个微分方程的模型，就必须写一段程序，而且如果需要改变参数，则程序会更复杂。在 Simulink 出现后，由于它提供了友好的图形用户界面(GUI—Graphical User Interface)，用户只要进行简单的点击和拖动鼠标就能完成建模，就像使用铅笔在纸上画模型一样自如、方便。模型由 Simulink 标准库模块(也可以是用户自定义的模块)和连线组成。我们可以做个形象的比较，从物理意义上讲，每一个库模块就像是一个多功能硬件，而连线就相当于硬件之间的各种导线，这样整个模型看起来如同工程上使用的方框图一样，简洁、明了；从数学角度上看，一个库模块就相当于一个函数，所以也可将库模块称为 Simulink 库函数。具有不同功能或函数运算的 Simulink 库模块(或库函数)就组成了 Simulink 模块库(或函数库)。按工程使用的情况，库模块又分为若干类，如接收器模块、输入源模块、线性或非线性组件模块以及连接器模块等等，当然用户还可以根据需要定制和创建自己的模块。

2. 易于进行模型分析

由于模型是层次型的，因此建立模型的过程就如同通常的设计过程那样，可以是自上而下亦或自下而上。你可以先在系统的高层次上分析、研究，然后双击具体模块或子系统图标，进一步分析模型下一层的细节。这种方法便于分析模型的组织结构以及各部分的相互关系。

3. 优越的仿真性能

在定义好一个模型之后，用户就可以通过Simulink菜单或者在MATLAB命令窗口中输入命令这两种方式，选择一种集成方法来进行仿真。前者特别适合于交互式工作，而后者则在进行仿真的批处理（例如进行蒙特卡罗仿真或者需要一个参量在一定范围内连续改变数值）时非常有用。使用示波器或其他显示模块，可以在仿真运行的同时看到仿真的结果，若改变参数后运行就可立即看到相应的结果，这适用于因果关系的问题研究。当然，仿真的结果也可以送入MATLAB工作空间以便做进一步的处理。模型分析工具为线性化工具和整理工具（可以通过输入MATLAB命令得到），MATLAB的所有工具以及Simulink本身的应用工具箱都包含这些工具。由于MATLAB和Simulink是集成在一起的，所以可以在任何一个环境的任意节点上进行仿真、分析和修改模型。

需要指出的是，虽然几乎所有MATLAB应用程序工具箱都包含有体现Simulink模块化设计的本专业模块，但在实际应用中，也经常需要与本书中介绍的模块结合起来使用。

注意：Simulink不能脱离MATLAB而独立工作。

1.2 应用程序工具箱介绍

我们知道，Simulink是建立在MATLAB的基础之上的。Simulink的用户可以直接利用涉及众多领域的MATLAB工具来创建、分析和优化系统。在这些工具中，最突出的就是MATLAB应用工具箱，这是一组专门收集和汇总的M文件（用MATALB语言编写的文件），用于处理某些特殊类型的问题。工具箱并不仅仅是一些有用函数的集合，在某种意义上它们代表着世界顶级研究人员在各自领域所做出的努力。也因为如此，随着MATLAB版本的升级，其内容也会有相应的调整。我们真诚希望，能在未来升级的版本中看到读者所提供的新算法、新理论。

Simulink的所有工具箱都是使用MATLAB语言制作的，因此有几点说明提请读者注意：

(1) 虽然如软件所说每个工具箱都是建立在可靠的数字、稳定的精确性和MATLAB多年的经验之上的，但在使用时，还应特别注意适用范围。

(2) 在工具箱与Simulink或者任何其他可供使用的工具箱之间可以实现即时的无缝集成。

(3) 可以充分利用MATLAB的开放特性：可以检查M文件，编辑M文件的内容，或者以M文件为模板创建用户定义的函数。

(4) 每个工具箱在任何能运行MATLAB的计算机平台上都可以使用。

下面列出了目前MathWorks提供的专业工具箱的名称及简单说明。由于篇幅的限制，虽然我们不可能在本书中对它们进行详细的介绍，但是，我们相信读者学完本书后，只要掌握了标准模块的使用，模型的建立方法，再使用专业工具箱的模块就绝对不存在技术上的障碍。

通信工具箱(Communications Toolbox)。通信工具箱提供一系列的集成工具，可以加速现代通信系统的设计、分析及仿真过程。它将MATLAB高级语言与便于使用的Simulink模块图界面相结合，为通信工程师提供了强大、全面的通信系统设计和分析功能。同时，这个工具箱对远程通信、电话服务、航天以及计算机外围设备等不同产业同样有用。

控制系统工具箱(Control System Toolbox)。控制系统工具箱作为MATLAB控制设计工

具箱家族的基础,包含了自动控制系统建模、分析和设计的各种函数。随着计算机和各类传感器价格的下降,自动控制系统的应用日益增长,使自动控制器不仅应用于汽车、航天器系统、计算机外围装置以及过程控制等各种高技术设备,并且在家用电器上也得到了应用。

财务工具箱(Financial Toolbox)。在 MATLAB 下运行的财务工具箱提供了一系列丰富实用的用于财务及数量分析的财务函数,主要功能包括证券定价、利息和收益率计算、偏差分析和业务优化等。财务工具箱的使用需要有统计和优化工具箱的支持。将 Simulink 的图形界面引入蒙特卡罗和非随机仿真,有利于对固定收入有价证券、转价证券以及其他证券定价。

频域系统辨识工具箱(Frequency-Domain System Identification Toolbox)。频域系统辨识工具箱是一组 M 文件,用于在测量线性系统频率响应的基础上建立系统的模型。

模糊逻辑工具箱(Fuzzy Logic Toolbox)。模糊逻辑工具箱提供一整套基于 GUI 的工具,用于设计、仿真和分析模糊推理系统。这个工具箱可以使用户利用既易于理解又十分有效的方法,按照自然语言的特定规则和相互关系,将一输入空间映射到另一具有任意复杂性的输出空间。系统可以在 MATLAB 中实现仿真,也可以形成一个 Simulink 模块,生成代码并独立执行。

高阶谱分析工具箱(Higher-Order Spectral Analysis Toolbox)。这是利用高阶谱进行信号处理的工具,对于分析来源于非线性过程的信号或者受到非高斯噪声干扰的信号特别有用。

图像处理工具箱(Image Processing Toolbox)。图像处理工具箱提供了进行图像处理和算法开发的工具。它包含的工具有过滤设计、图像修复、图像增强、分析与统计、颜色、几何与形态操作以及二维变换等。

LMI 控制工具箱(LMI Control Toolbox)。LMI 是一种特别的凸面优化问题,在包括控制、辨识、筛选、结构的设计、图形理论和线性代数在内的许多领域中出现。LMI 控制工具箱的另一个显著特点是它包含了基于 LMI 的用于控制系统设计的多种工具,并且包括诸如鲁棒稳定性和性能分析、鲁棒增益设计以及融 H 无限大、LQG 与极点为一体的多目标控制器设计等方面的应用。

模型预测控制工具箱(Model Predictive Control Toolbox)。模型预测控制工具箱特别适用于那些有许多输入和输出变量,并且大部分变量都附带限制条件的控制应用系统。应用于化学工程及其他过程控制方面。

μ 分析与综合工具箱(Mu-Analysis and Synthesis Toolbox)。分析与综合工具箱包括 H∞ 优化控制、分析与综合等专门工具,提供了多变量线性系统高级鲁棒控制设计的途径。

NAG 基础工具箱(NAG Foundation Toolbox)。NAG 基础工具箱包括 200 多个来自 NAG Fortran 语言子程序库的数值计算函数,为边界值问题、优化、自适应积分、面和曲线的拟合以及其他应用提供了专门的工具。

神经网络工具箱(Neural Network Toolbox)。神经网络工具箱是一组用于神经网络设计和仿真的 MATLAB 函数集。神经网络是受到生物神经系统的启发而建立的一种计算体系,它适用于一般的分析方法极端困难或者根本不可能的场合,比如模式识别、非线性系统辨识和控制等。

优化工具箱(Optimization Toolbox)。优化工具箱提供了常规的线性和非线性函数的优化命令,包括带有限制条件的函数。一个优化问题可以形象化地看作是在一个复杂的、以等高线作标记的高地上寻找最低(或最高)点的问题。

偏微分方程工具箱(Partial Differential Equation Toolbox)。偏微分方程(PDE—Partial Differential Equation)工具箱将研究和解决 PDE 问题的 MATLAB 技术计算环境扩展到了二维空间(2-D)和时间域。这个工具箱提供了一组命令函数和直观的图形用户界面,用于使用有限元法(FEM—Finite Element Method)来预处理、求解和后处理普通的二维偏微分方程。偏微分方程工具箱还提供了自动与自适应的拟合功能以及常见的八种偏微分方程应用模型。这些应用在工程学和物理学中都是最常见的,如热传递、结构力学、静电学、静磁学和扩散等。

QFT 控制设计工具箱(QFT Control Design Toolbox)。量化反馈理论(QFT—Quantitative Feedback Theory)是一种用来为不确定系统设计控制器的频域方法,可以直接反映控制器的复杂性(这实际上代表了实现起来的难易程度)及其特性之间的平衡关系。

鲁棒控制工具箱(Robust Control Toolbox)。鲁棒控制工具箱提供了一系列专门的工具以分析和综合控制系统,这些系统对于可能在真实世界出现的不确定性而言具有鲁棒性。

信号处理工具箱(Signal Processing Toolbox)。信号处理工具箱包括各种信号处理工具,可应用于音频(如 CD 和数字录音带)、视频(数字 HDTV、图像处理和压缩)、远程通信(传真和电话)、医学(CAT 扫描、磁共振成像)、地球物理学和经济计量学等领域。

样条工具箱(Spline Toolbox)。样条工具箱提供了一组 M 文件以构造和使用分段多项式逼近的样条。由于可以用样条来逼近其他函数的值,从而避免了其他逼近方法(如分段线性曲线逼近)可能带来的不利影响,所以样条在数值逼近方面十分有用。

统计工具箱(Statistics Toolbox)。统计工具箱提供了一组 M 文件用于统计数据的分析、建模和蒙特卡罗仿真,另外有相应的 GUI 工具,提供一些统计学和概率论基本概念的内容,如多种随机信号的产生等。

符号数学工具箱(Symbolic Math Toolbox)。符号数学工具箱为 MATLAB 增加了一组以 Maple V 为基础的用于符号计算和变量精度算法的集成工具。扩展的符号数学工具箱支持 Maple 程序设计并提供了附加的特殊函数。

系统辨识工具箱(System Identification Toolbox)。系统辨识工具箱是一个用于判断和辨识的工具集。系统辨识是指根据一个物理系统的输入和输出寻求该系统的数学模型。

小波工具箱(Wavelet Toolbox)。小波工具箱提供了一个用于检查局部的、多尺度的或非固定现象的综合子程序集。凡是用到傅里叶技术的地方,都可以利用小波方法提高理解问题的程度和分析问题的性能。小波工具箱可应用于信号处理和图像处理等诸多方面,如话音和音频处理、通信、地球物理学、金融以及医学等。

1.3 Simulink 实时工作间

Simulink 实时工作间(Real-Time Workshop)能够直接从 Simulink 模块图自动生成 C 或 C++语言源程序代码,这样就可以使连续时间、离散时间或混合系统模型能够在多种计算机平台乃至硬件上执行。它必须在 Simulink 环境下运行。

1.3.1 用途

实时工作间可以用于下列目的:

(1) 快速原型化。作为一种快速的原型化工具,实时工作间可以迅速实现用户的设计思

想而无需冗长的手工编码与调试过程。通过开发图形化的 Simulink 模块图并自动生成 C 语言源程序代码，可以实现控制、信号处理和动态系统算法。

(2) 嵌入式实时控制。如果使用 Simulink 来设计一个系统，就可生成用于实时控制器或数字信号处理器的代码，并进行交叉编译、链接和嵌入到所选定的目标处理器中。实时工作间支持 DSP 板、嵌入式控制器以及各种个人和商用硬件。

(3) 实时仿真。可以根据硬件的在线仿真创建和执行整个系统或特定子系统的源程序。这一方面的典型应用包括训练模拟器(飞行员在位仿真)、实时模型有效性验证以及测试等。

(4) 独立仿真。可以直接在主机上运行，也可以传输到其他系统实现远程运行。由于时间历史记录是以二进制或 ASCII 码文件保存在 MATLAB 中的，因此就能很容易地载入 MATLAB 中用于随后的分析或图形显示等。

1.3.2 主要特性

由于实时工作间有一系列好的特性和能力，这使其能够灵活地运用于各个方面，包括：

(1) 自动生成程序代码以处理连续时间、离散时间以及混合时间系统。

(2) 优化代码以保证其快速执行。

(3) 控制构架应用程序接口(API—Application Program Interface)利用用户化的 M 文件来自动建造目标文件并将其移植到所选硬件上。

(4) 具有移植性的代码可以用于多种环境。

(5) 简洁、易读、注释详细的代码使得维护更加方便。

(6) 从 Simulink 到外部硬件的交互式参数下载使得系统可以随意转换。

(7) 由菜单驱动的图形化用户界面使得软件使用起来非常容易。

1.3.3 运行环境

实时工作间支持下列目标环境：

(1) 使用 TI C30/C31/C40 DSP 的 dSPACE DS1102,DS1002 和 DS1003。

(2) VxWorks,VME/68040。

(3) 具有 Xycom,矩阵、数据编译或输入/输出设备和 Quanser Multiq 板的基于 486 或以上的个人计算机系统。

1.4 Ada 扩展的实时工作间

Ada 扩展的 Simulink 实时工作间(RTW—Real-Time Workshop)能够直接从 Simulink 模块图自动生成 Ada 语言源程序代码。这就可以使连续时间、离散时间或混合系统模型能够在多种计算机平台乃至实时硬件上执行。它必须在 Simulink 环境下运行，其用途和特性与 RTW 类似，只是它所生成的代码是 Ada 语言而非 C 或 C++语言。

Ada 扩展的 RTW 为下列 Ada 83 编译器提供关键的支持：

(1) UNIX 平台的 Rational VADS。

(2) 微软 Windows 专业版的 Thomson ActivAda。

(3) Windows NT 的 Thomson ActivAda。

1.5　模块集

与 MATLAB 及其应用工具箱类似，MathWorks 提供了与 Simulink 一起使用的模块集。所谓模块集，实际上就是一组 Simulink 模块，是 Simulink 主库中的独立程序库的集合。需要注意：所有模块集只能在支持 Simulink 和 MATLAB 的 Windows 或 UNIX 操作系统下运行。

1.5.1　DSP 模块集

DSP 模块集必须在 Simulink 环境下运行。它将 Simulink 的用途扩展到用于基于 DSP 的设备及系统的快速设计和仿真，它给 Simulink 提供了用于信号处理算法交互式模块图仿真与评估的直观工具。同样，该模块集的图形化编程环境也使得创建、修改和原型化 DSP 设计更加容易。DSP 模块集的应用包括通信系统、计算机外围设备、语音和音频处理、汽车和航天器控制，以及医学电子设备的设计和分析。在解决时域和频域算法，包括自适应噪声消除等方面的问题时，是较为理想的工具。

1.5.2　定点模块集

定点模块集只能在 Simulink 环境下运行。它包含一组扩展了 Simulink 标准模块库的模块图组件。利用这些模块组件，用户可以创建采用定点算法的离散时间动态系统。因此，Simulink 可以对在定点系统中经常遇到的诸如控制系统和时域滤波等实际问题和结果进行仿真。

定点模块集提供的模块可用于下列场合：

(1) 加法和减法；

(2) 乘法和除法；

(3) 求和；

(4) 增益和常数；

(5) 浮点与定点信号间的转换；

(6) 一维和二维查表；

(7) 逻辑运算符；

(8) 关系运算符；

(9) 定点信号变换/饱和；

(10) 两个数值间的切换；

(11) 延迟；

(12) 反 Delta 运算符；

(13) 监视信号。

几点说明：

(1) 定点模块集中的信号变换模块用来进行浮点信号和定点信号之间的转换，构建由 Simulink 标准模块库组件与定点模块组成的 Simulink 模块图。例如，可以利用 Simulink 标准模块库建构模型，然后用定点模块模拟控制器。数据量程模块可以提供仿真期间在模块图中的任意一点所能遇到的最大值和最小值。

(2) 定点模块集允许用户使用无符号的或 8 位、16 位、32 位字长的数据格式构建模型。

在同一个模块图中可以使用具有不同字长的模块。可以通过在定点模块内指定二进制点的位置对定点值进行标定。在仿真过程中，数据类型可以更改，可以即时看到不同的字长，不同的二进制点位置，不同的截位，不同溢出检验等所产生的结果。

(3) 该模块集的另一个特点是能对给出二进制的非溢出最大精度值点的自动定位。利用数据量程模块，可以将二进制点确定在合适的数值上。

1.5.3 非线性控制设计模块集

非线性控制设计(NCD—Nonlinear Control Design)模块集提供基于时域的鲁棒非线性控制设计。控制器设计可以开发成 Simulink 中的模块图。用户选择一组可调的模型参数，并将时间响应约束图形化地加在选定的输出信号上。自动地应用连续的仿真和优化方法，从而调整所选择的模型参数。

1.5.4 电力系统模块集

电力系统模块集可以使科学家和工程师构建电力系统的仿真模型。在 Simulink 环境下，模块集使用点击和拖动操作就可以完成模型的构建，不但可以迅速地画出电路拓扑结构图，而且可以分析电路与机械、热、控制以及其他学科的相互作用，这是因为在 Simulink 仿真的所有电路部分都与 Simulink 扩展模型库发生了相互联系。由于 Simulink 是以 MATLAB 作为计算引擎，所以 MATLAB 的工具箱也可以供设计者使用。

这个模块集包含典型电力设备的模型，如变压器、线路、机器以及电力电子学等等。这些模型是建立在 Hydro-Quebec 电力系统测试和仿真实验室的实践经验基础之上的。

1.6 模型演示

Simulink 提供了很多有趣的演示程序。在了解了一些基础知识以后，我们不妨试着运行一个演示模型，看一看，感受一下。

1.6.1 运行演示模型

通过一个模拟房子的热力学问题的模型(图 1.1)，希望读者能很快地学会运行所有的演示程序，从中体验 Simulink 带给我们的快乐。运行步骤如下：

(1) 启动 MATLAB。

(2) 在 MATLAB 命令窗口键入 thermo，运行演示模型。这个命令将自动启动 Simulink 并创建一个包含该模型的窗口。当打开该模型时，Simulink 将打开一个示波器模块，该模块有两个显示屏，分别标记着 Indoor vs. Outdoor Temp 以及 Heat Cost($)。

(3) 下拉出 Simulation 菜单并选择 Start 命令(或者在微软的 Windows 中按下 Simulink 工具栏上的 Start 按钮)，即可开始运行仿真。随着仿真的运行，户内和户外的温度将显示在 Indoor vs. Outdoor Temp 图上，而累计的取暖费用则显示在 Heat Cost($)图上。

(4) 在 Simulation 菜单中选择 Stop 命令(或按下工具栏上的 Pause 按钮)，即可停止仿真。如果想要了解这个模型的其他情况，请查阅 1.6.3 节。

(5) 运行完后，可选择 File 菜单中的 Close 命令关闭仿真模型。

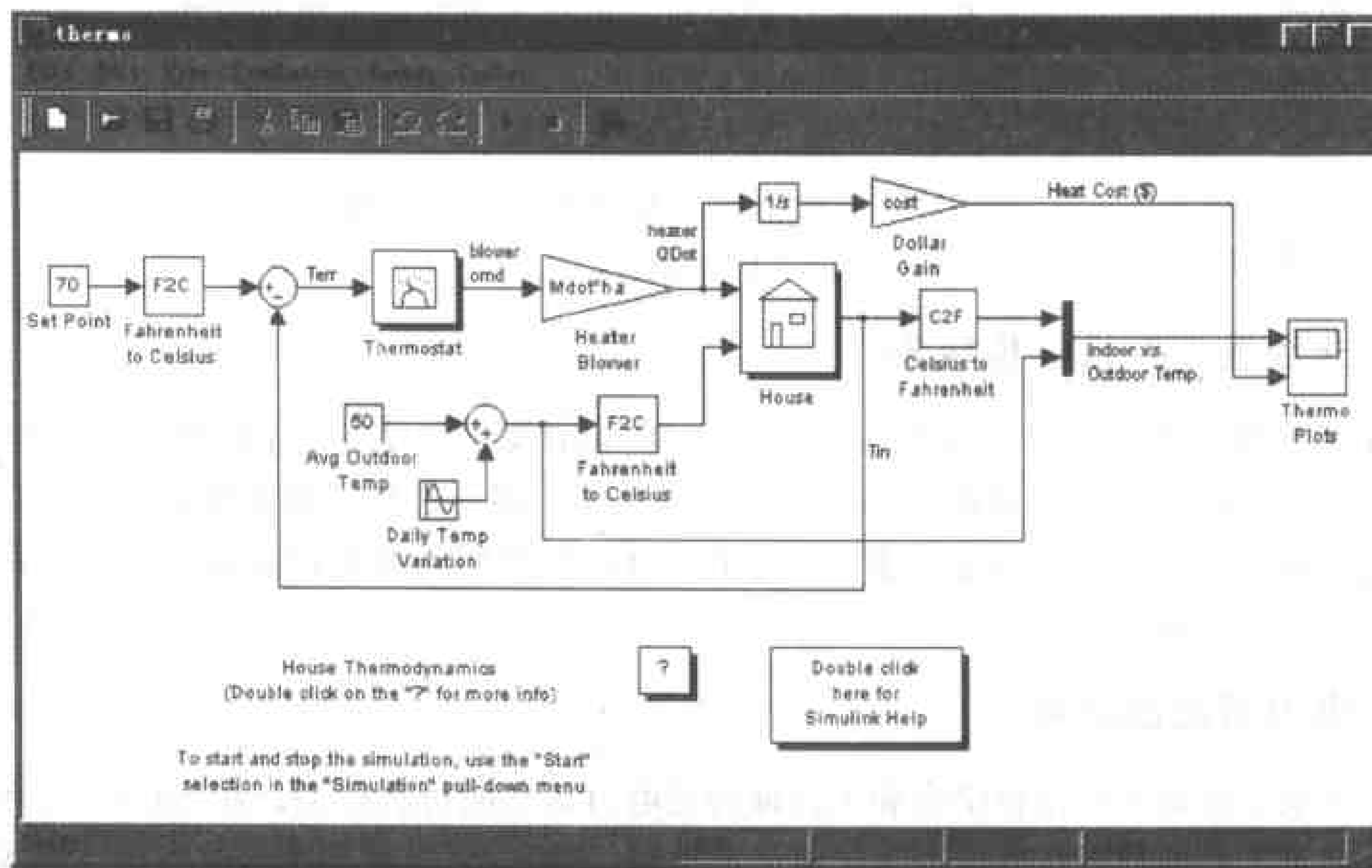


图 1.1　一个 thermo 模型的模块图

1.6.2　演示模型描述

这个演示程序用一个简单的模型模拟了一间房子的热力学问题。自动调温器被设置为 70℉并受室外温度的影响。室外温度随一个振幅为 15℉、基准温度为 50℉的正弦波变化。这样就仿真了一天中温度的变化情况。

本模型使用子系统简化了模型框图，并创建了可重复使用的系统。所谓子系统，就是用一个 Subsystem 模块来表示的一组模块。本模型包含了五个子系统：一个是 Thermostat(自动调温器)，一个是 House(房间)，其余三个是 Temp Convert(温度转换)子系统(其中两个将 Fahrenheit(华氏度)转换为 Celsius(摄氏度)，一个将摄氏温度转换为华氏度)。

室内和室外温度都送到房间子系统并更新室内温度。双击房间模块，可以看到子系统内

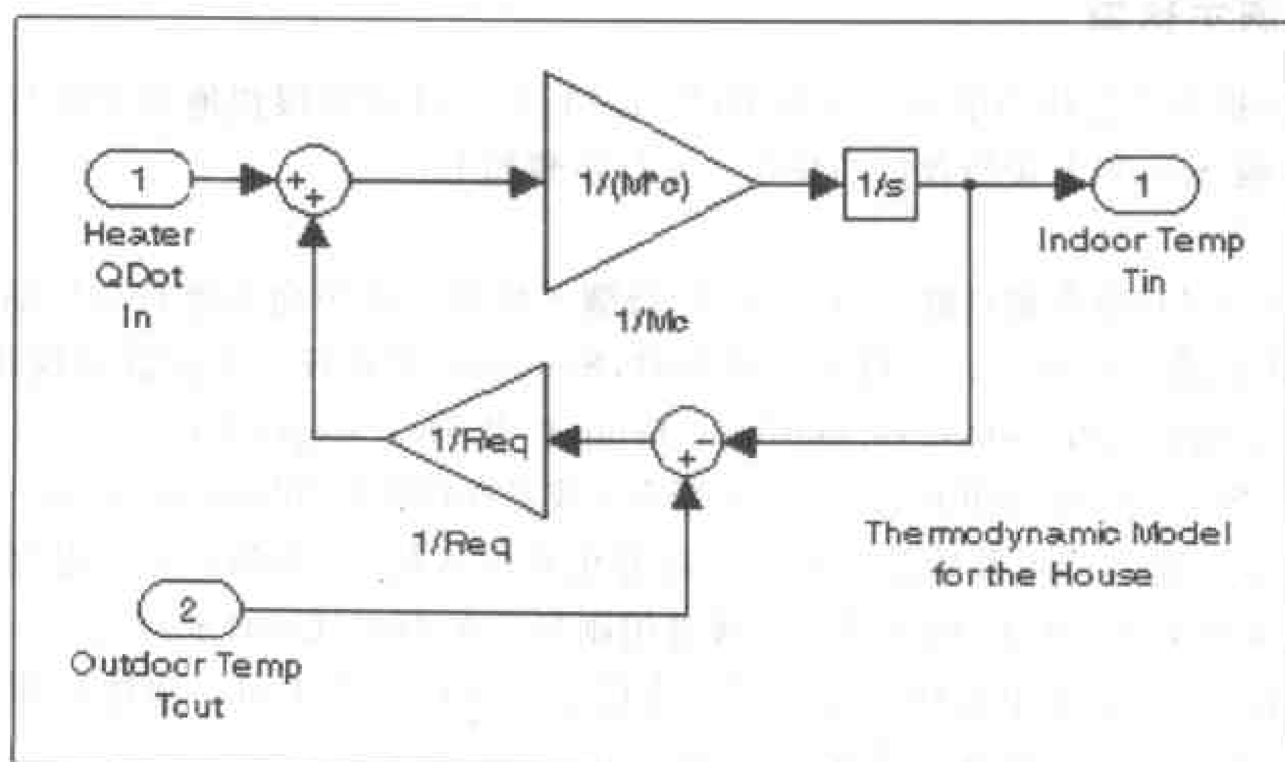


图 1.2　House(房间)子系统模型

部的详细构造(图 1.2)。

自动调温子系统模拟自动调温器的工作,确定打开和关闭加热系统的时间。双击该模块,可以看到子系统内部的详细结构(图 1.3)。

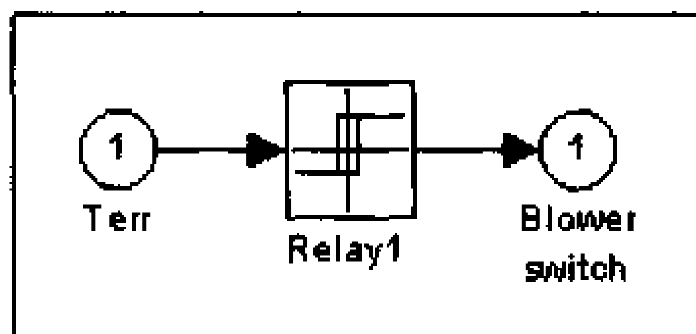


图 1.3　自动调温器子系统

室外和室内温度由相同的子系统从华氏温度转换为摄氏度,图 1.4。

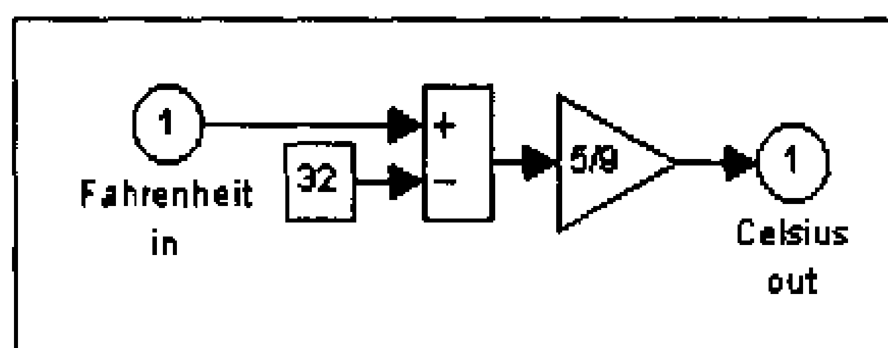


图 1.4　华氏度转换为摄氏度(F2C)

当加热系统打开时,取暖费用就被计算出来并显示在 Thermo Plots 示波器的 Heat Cost($)图上,室内温度则显示在 Indoor Temp 示波器上。

1.6.3　初试身手

看着不如做着,现在读者可以按照下面的提示试着做一做,看看该模型对不同的参数所做出的反应:

(1) 每个示波器模块都包含一个或多个信号显示区域和控制功能,你不妨移动鼠标,将光标指向不同区域,然后点击,就可看见诸如显示信号的范围发生变化,信号的某一感兴趣部分被放大,或者执行其他任务。水平轴代表时间,垂直轴代表信号量(幅)值。有关示波器模块的更多信息,请参考第 5 章中有关"Scope 模块"的内容。

(2) 标记为 Set Point 的常数模块(在模型的左上角),可以设定所需要的室内温度。在仿真运行时,打开此模块,将温度值设置为 80℉,看看室内温度和取暖费用如何变化。另外,调整一下室外温度(Avg Outdoor Temp 模块),看看它对仿真有何影响。

(3) 打开标记为 Daily Temp Variation 的正弦波(Sine Wave)模块并改变振幅参数,可以调整日常温度变化量。

1.6.4　关于本演示程序的进一步探讨

本演示程序反映了在构建模型过程中,通常要完成的几项任务:

(1) 运行仿真的过程,包括指定参数以及利用 Start 命令启动仿真,详见第 5 章。

（2）可以将复杂的、相关联的模块封装在一个模块中，组成一个子系统。有关创建子系统的内容详见第3章。

（3）利用封装特性，可以为一个模块创建一个定制图标并设计一个对话框，详见第6章。在 thermo 模型中，所有的 Subsystem 模块都有利用封装特性创建的定制图标。

（4）Scope（示波器）模块如同一个真实的示波器一样可以显示输出波形，它也是最常用的模块之一。有关 Scope 模块的详细内容参见第5章。

1.6.5 其他演示程序

除了上面的例子外，系统还提供一些其他的演示程序，进一步举例说明了建模概念。读者可以通过 Simulink 模块库窗口访问这些演示教程：

（1）在 MATLAB 命令窗口键入命令：Simulink3，就会出现 Simulink 模块库窗口（图1.5）。

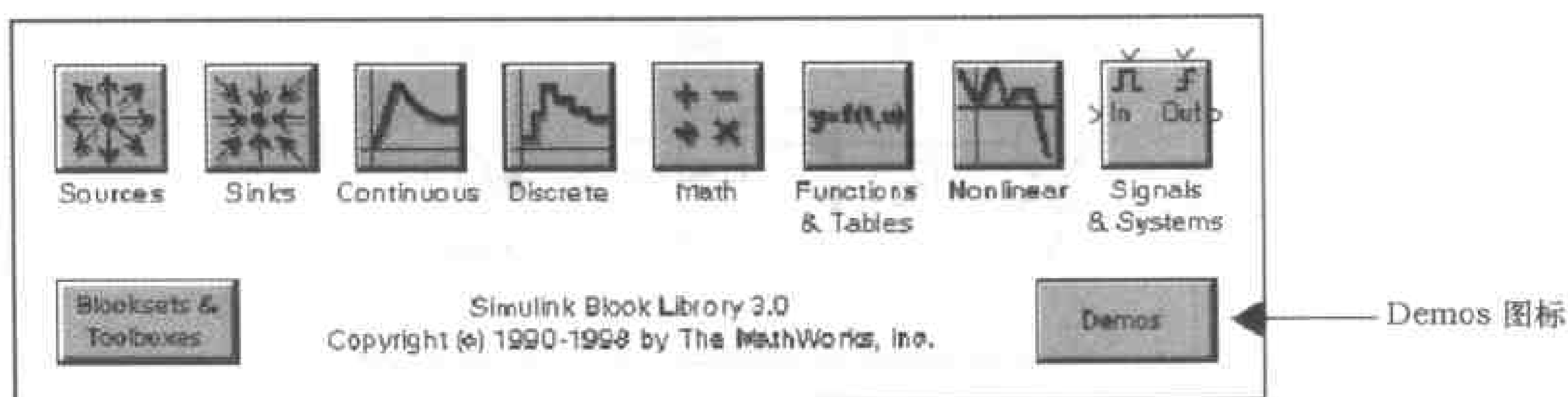


图1.5 Simulink 模块库窗口

（2）双击 Demos 图标，MATLAB 的演示程序窗口就会出现。这个窗口包括几个说明 Simulink 特性的有趣示例。

1.7 创建一个简单模型

通过本节，读者将学会如何使用常用的建模步骤来建造一个自己的模型的方法。构建模型在本节只做简单的介绍，更具体的内容将在以后的章节中详细地介绍。

现在我们构建对一个正弦波进行积分运算的模型，并显示该正弦波及其积分结果，其模型如图1.6所示。

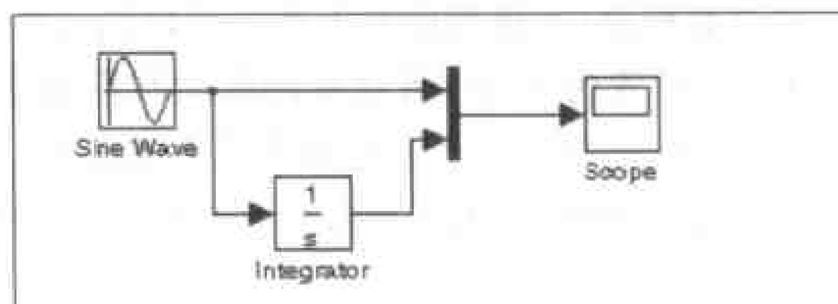


图1.6 一个简单的模型

建模步骤：要创建这个模型，按照如下步骤进行。

步骤 1：进入 Simulink 系统

首先在 MATLAB 命令窗口中键入 Simulink，或者用鼠标左键单击（约定：今后如未加说明，则单（或双）击表示用鼠标左键单（或双）击）Simulink 图标 。在微软 Windows 操作系统下，就会出现如图 1.7 所示的 Simulink 模块库浏览器窗口。



图 1.7　Simulink 模块库浏览器窗口

在 UNIX 系统下，Simulink 库窗口的形式如图 1.8 所示。

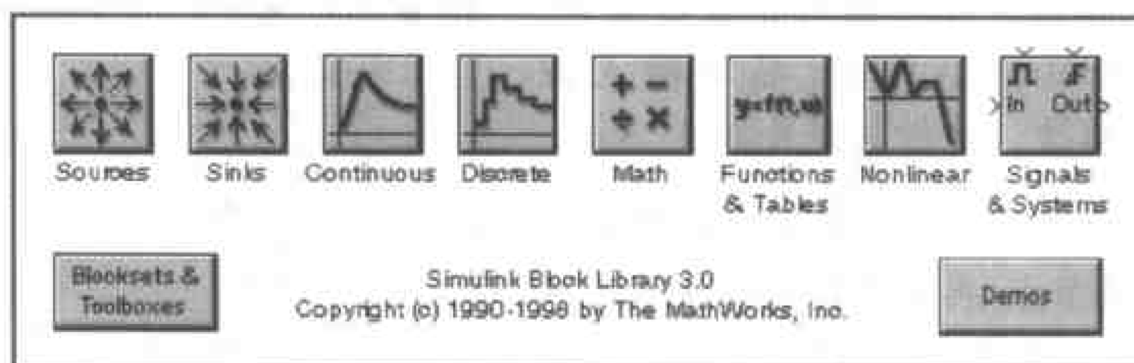


图 1.8　UNIX 系统下的 Simulink 库窗口

步骤 2：建立新模型窗口

在 UNIX 系统下创建一个新模型，是从 Simulink 库窗口的 File 菜单下，选择 New 子菜单中的 Model 命令。在 Windows 操作系统下创建新模型，只须单击模块库浏览器工具栏的 New Model 按钮即可，如图 1.9 所示。



图 1.9　Windows 系统下，新建模型按钮位置

单击 New Model 按钮后，Simulink 则打开一个名为 untitled1 的新建模型窗口，如图 1.10 所示。如果需要，可以把这个新模型窗口拖到屏幕右边，以便同时看到它和模块库的内容。

步骤 3：选取模块

要创建这个模型，接下来就需要选择库模块，可以从下面的 Simulink 模块库中找到相应的模块，并将它们拷贝到模型中：

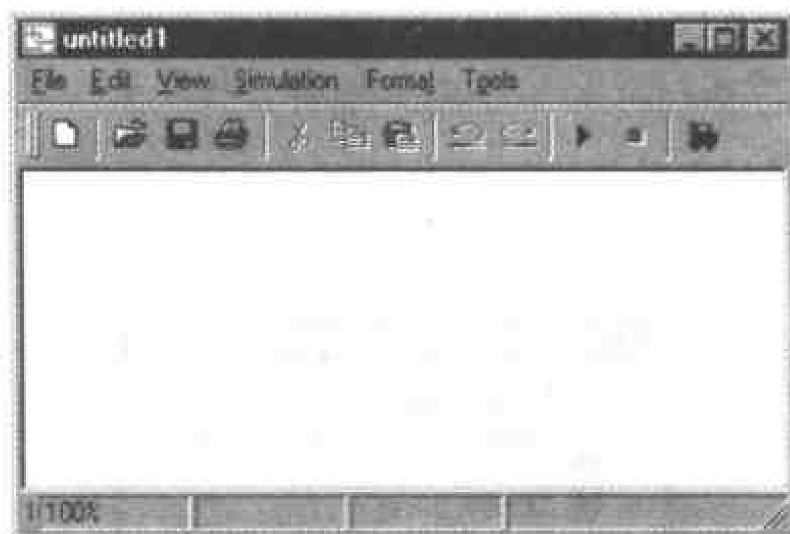


图 1.10 新建(空白)模型窗口

(1) 输入源(Sources)模块库中的 Sine Wave 模块;

(2) 接收器(Sinks)模块库中的 Scope 模块;

(3) 连续系统(Continuous)模块库中的 Integrator 模块;

(4) 信号与系统(Signals & Systems)模块库中的 Mux 模块。

这里仅以 Sine Wave 模块的选取过程为例说明。在 Windows 操作系统下,可以通过库浏览器(图 1.9),从输入源模块库中拷贝正弦波模块。首先应展开库浏览器结构树,显示输入源(Sources)库中的模块。其做法是先单击 Simulink 库的节点 ⊞,使其变为 ⊟,显示出包括输入源(Sources)在内的节点,再单击输入源库节点显示出下一级输入源库模块节点,最后单击正弦波节点(Sine Wave)就可选中 Sine Wave 模块。这之后,库浏览器窗口就会如图 1.11 所示。



图 1.11 选中 Sine Wave 后的情况

接下来的工作就是拖动浏览器中的正弦波节点(按住鼠标左键),把它放到新建模型窗口内(释放鼠标),Simulink 则在你放下节点图标的位置创建正弦波模块的副本。在拖动鼠标的过程中,应看到节点图标随光标移动。

如果是 UNIX 系统,则需要从输入源库窗口中拷贝正弦波模块,应在 Simulink 库窗口(如图 1.8 所示)中双击输入源(Sources)图标,就可打开输入源库窗口(如图 1.12 所示)。当然,在 Windows 操作系统下,也可以通过先在库浏览器上右键单击 Simulink 节点,然后单击 Open the 'Simulink' Library 按钮,打开 Simulink 库窗口。

将正弦波模块从源窗口拖到刚才新建模型窗口中,如图 1.13 所示。

用同样的方法把其他模块从它们各自所在的库中拷贝到模型窗口上。用拖动模块的方法,可以随意把模块在模型窗口内从一个地方拖到另一个地方。也可以采用选中模块然后按动键盘上的箭头键的方法在短距离内移动模块。在把所有的模块都复制到模型窗口上后,应该可以看到如图 1.14 所示的情形。

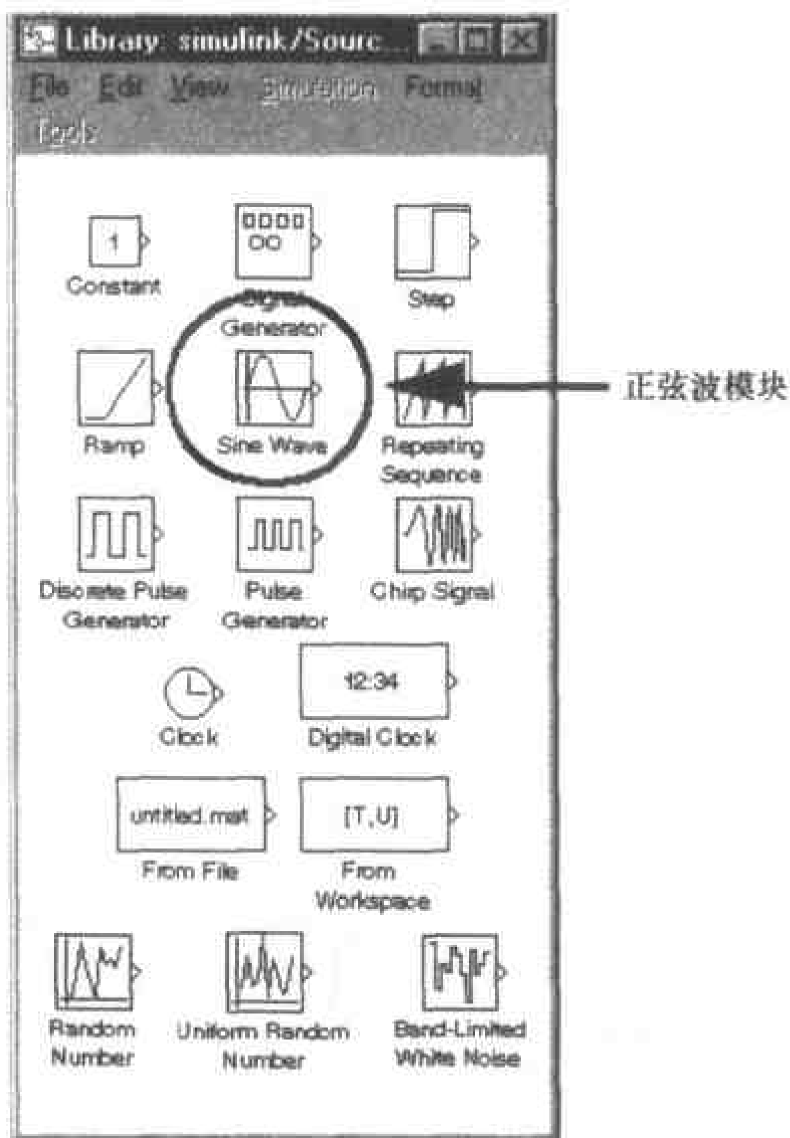


图 1.12 输入源库及包含的模块



图 1.13 新建模型及正弦波模块

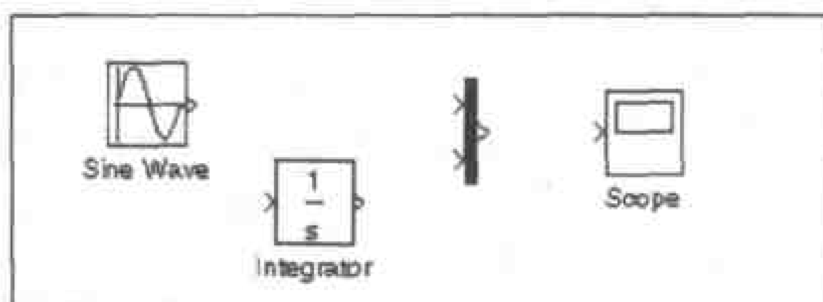


图 1.14 新建模型及所有模块

步骤 4:连接模块

观察模型中的模块图标就会发现:在正弦波模块的右侧有一个角括号“>”,在 Mux 模块的左侧有两个。这个“>”记号从模块中指出去,说明这里是一个输出端口;而该记号指向模块,则说明这里是一个输入端口,如图 1.15 所示。信号通常从一个输出端口输出,然后通过一根连接线输送到另一模块的输入端口。当这两个模块被连接起来之后,端口记号就会消失。

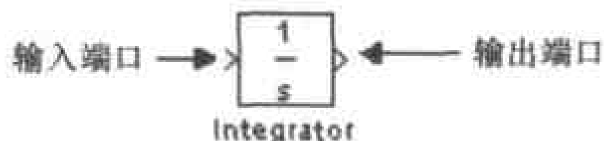


图 1.15 模块输入/输出端口示意

现在我们来连接模块。把正弦波模块与 Mux 模块上端的输入端口连接在一起。将光标放在正弦波模块右侧的输出端口上，注意此时光标形状应变成十字交叉状，如图 1.16 所示。

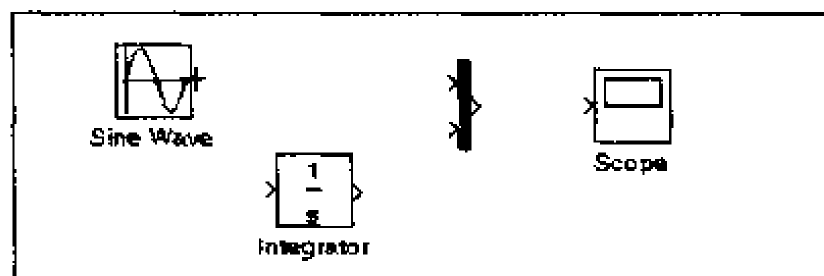


图 1.16 连接两个模块示意(1)

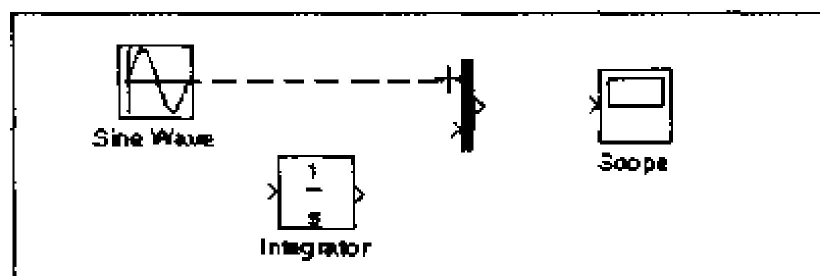


图 1.17 连接两个模块示意(2)

按下鼠标左键并将光标拖移到 Mux 模块上端的输入端口。注意，在按下鼠标键时，所画线段是虚线；当鼠标接近 Mux 模块时，光标形状变为双线十字交叉状(图 1.17)。

现在放开鼠标键，两个模块就连接在一起了。也可以在光标落在图标内时放开鼠标键，这样，这条连接线就被连接到离光标位置最近的输入端口(图 1.18)。

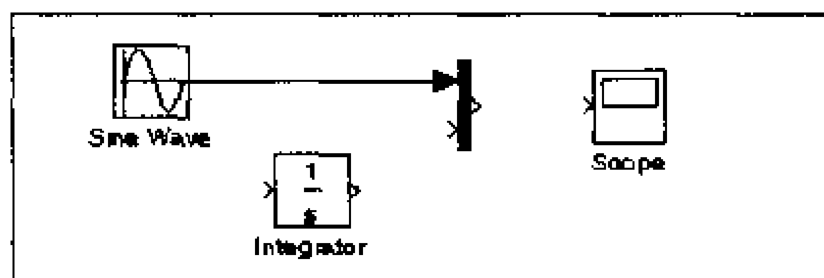


图 1.18 连接两个模块示意(3)

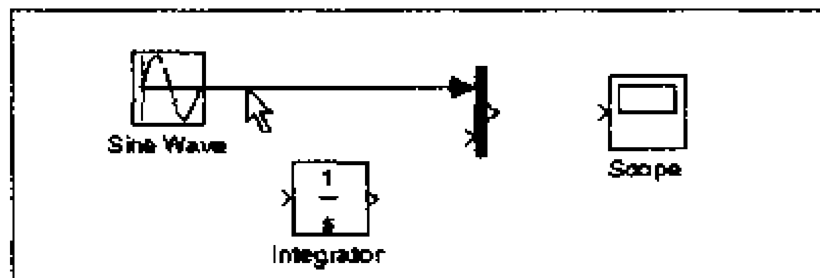


图 1.19 分支线连接示意(1)

这时如果读者翻到本节的开始再看一下那个模型(图 1.6)，就会发现大多数连线都连接着一个模块的输出端口和另一个模块的输入端口。然而有一条线却是将“一条线”同另一个模块连起来，也就是将正弦波(Sine Wave)模块的输出连接到积分器(Integrator)模块，这条线就叫做分支线，它所携带的也是从正弦波模块传输到 Mux 模块的信号。

画分支线与前面所述画其他线的方法略有不同。在已有的线上增加接点，可以进行下列步骤：

(1) 首先，将光标指在正弦波与 Mux 模块之间的连接线上(图 1.19)。

(2) 按下 Ctrl 键并不释放；按下鼠标键，然后将光标拖到积分器模块的输入端口(图 1.20)或者将光标拖到积分器模块之上(图 1.21)。

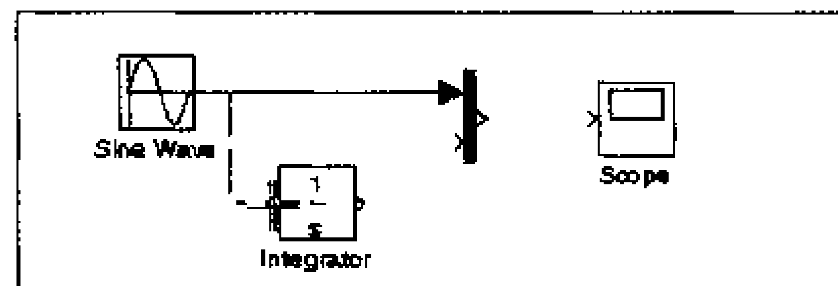


图 1.20 分支线连接示意(2)

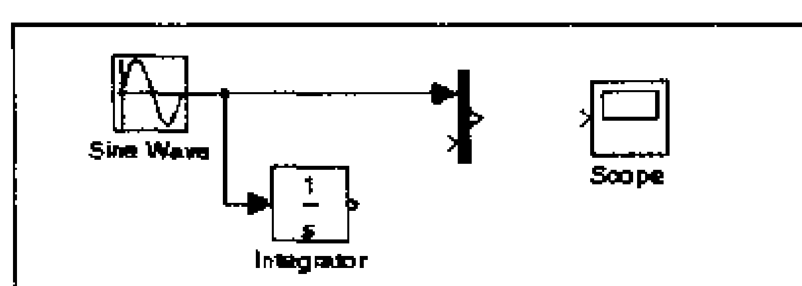


图 1.21 分支线连接示意(3)

(3) 释放鼠标键，Simulink 就在起始点到积分器模块的输入端口之间画出了一条连接线(图 1.21)。

当所有模块连接完成之后，模型应该如图 1.22 所示

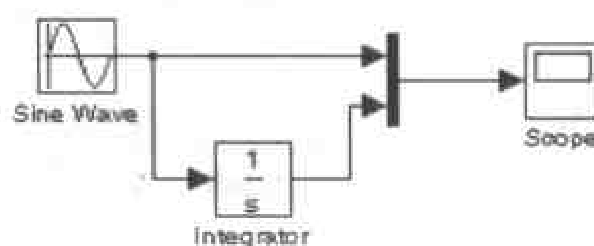


图 1.22 完全模型

步骤 5：仿真

现在，双击 Scope 模块图标打开示波器模块来观察仿真输出。保持示波器窗口处于打开状态，从 Simulation 下拉菜单上选中 Parameters，在接下来出现如图 1.23 所示的对话框中设置参数，在 Stop time(停止时间)栏内将 Simulink 运行仿真的时间设置为 10 s(默认值为 10.0 s)。

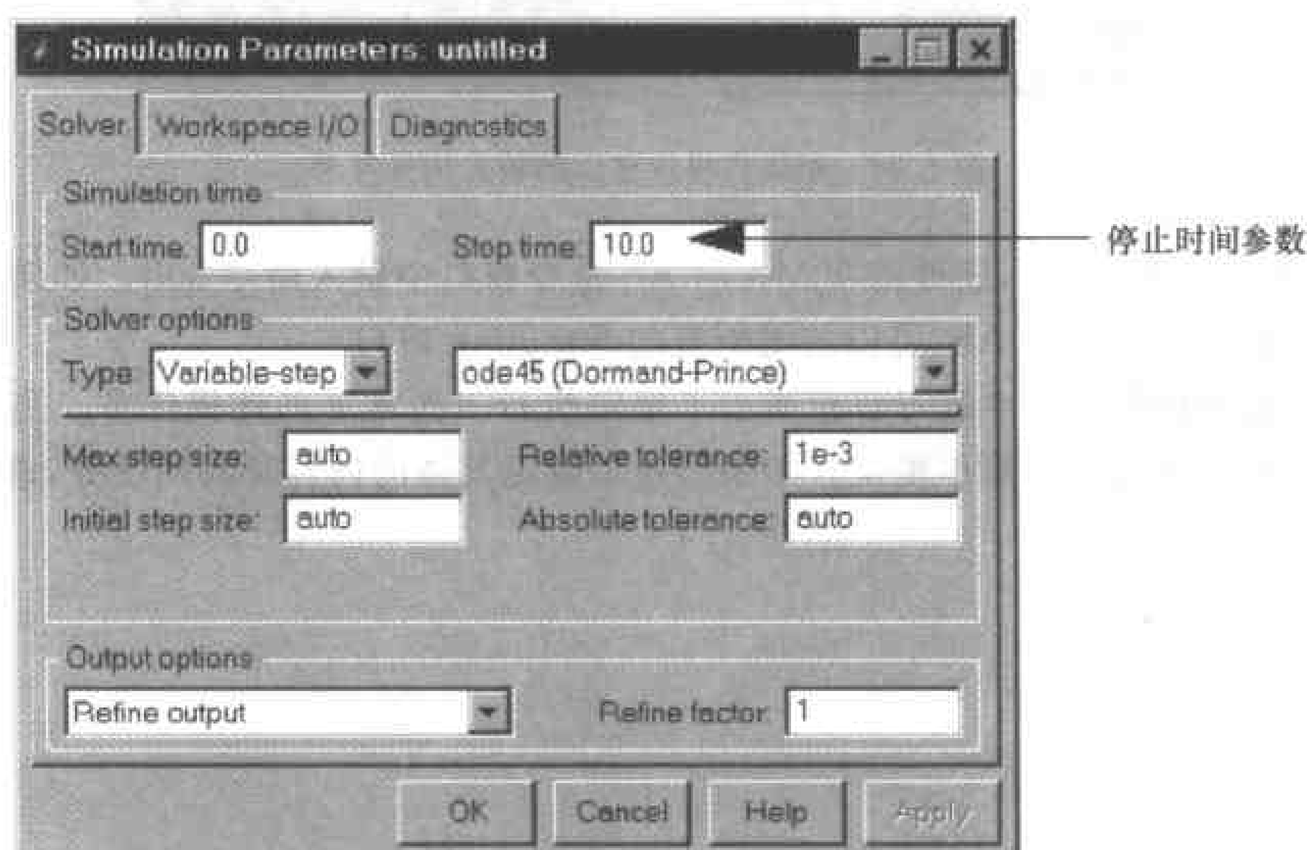


图 1.23 Simulink Parameters 对话框

单击 OK 按钮，关闭 Simulation Parameters 对话框。则 Simulink 保留设置该参数并关闭对话框。

从 Simulation 菜单中选择 Start，观察示波器模块的输入信号波形(图 1.24)。

当到达在 Simulation Parameters 对话框里设定的停止时间或者在 Simulation 菜单中选择 Stop 时，仿真即停止。

步骤 6：保存模型

在 File 菜单中选择 Save，在接下来出现的对话框中输入文件名和保存路径，然后确定，即可保存本模型。例如，我们用 SineInte 作为刚才构建好的模型名保存在 work 子目录(系统默认)下。

步骤 7：退出系统

要结束 Simulink 和 MATLAB，可选择 Exit MATLAB(在 Windows 系统)或者 Quit

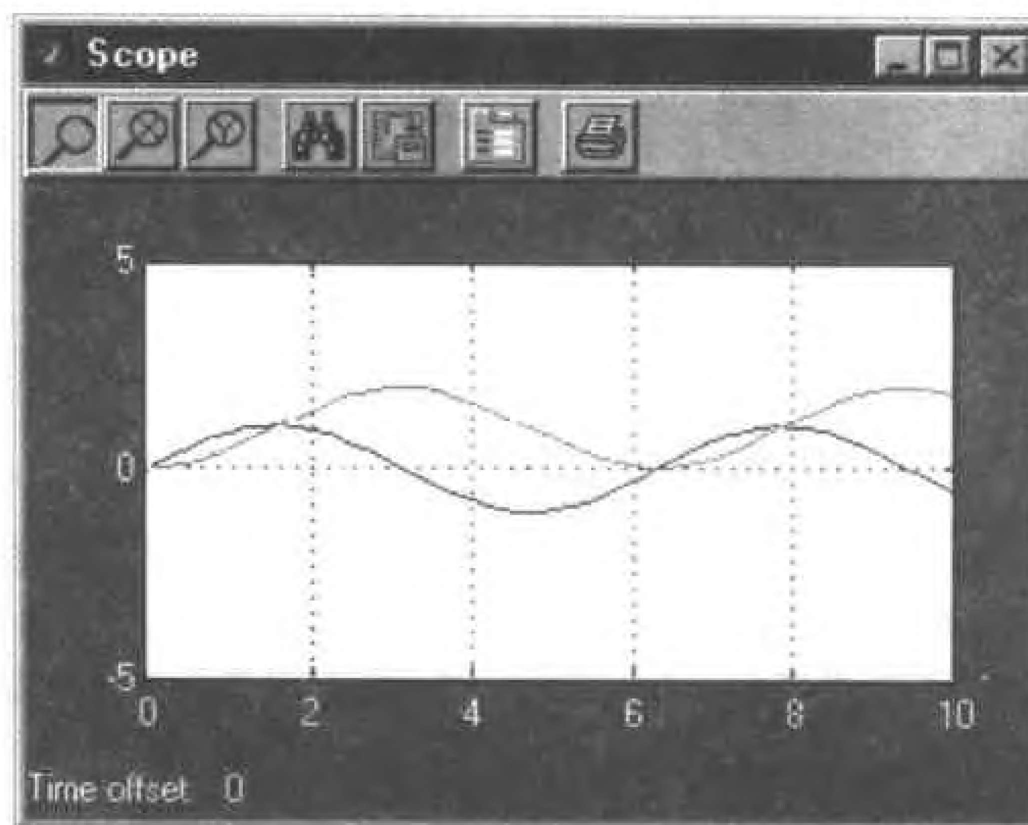


图 1.24 示波器模块显示的输入信号波形

MATLAB(在 UNIX 系统),还可以在 MATLAB 命令窗口中键入命令 quit。如果想离开 Simulink 但又不想关闭 MATLAB,可以关闭所有的 Simulink 窗口。

怎么样,是否很轻松?本练习仅仅演示了如何执行一些常见的建模任务。当然若读者愿意,可试着构建一些其他的简单模型。如果需要了解更具体的内容和其他任务,请阅读第 3 章的相关内容。

第2章 Simulink 系统操作命令

除了上一章介绍的利用 Simulink 提供的交互式方法进行模型设计外，利用命令来进行模型设计则更受一些编程员的喜爱，因为这更能反映出他们自身的综合才能而不必受交互式方法的局限。同时，了解和掌握本章的内容对更高层次的编程员来说也是必不可少的。

主要内容：本章介绍所有用于模型设计的命令或系统操作命令，并举例介绍。

学习目的：通过本章的学习，了解 Simulink 系统操作方法和模型设计命令，了解用命令设计模型的过程。

2.1 命令及要求概述

2.1.1 命令及其功能

表 2.1 给出了在本章中描述的所有命令及简要功能说明。

表 2.1　命令及其功能

命　令	功　　能
new_system	新建一个 Simulink 系统(模型)
open_system	打开一个存在的系统
close_system，bdclose	关闭一个系统
save_system	保存一个系统
find_system	寻找一个系统、模块、连线或注释
add_block	给一个系统添加一个模块
delete_block	从一个系统中删除一个模块
replace_block	替换系统内的一个模块
add_line	给一个系统添加一条连线
delete_line	从一个系统中删除一条连线
get_param	获取一个参数值
set_param	设置参数值
gcb	获取当前模块的路径名
gcs	获取当前系统的路径名
gcbh	获取当前模块的句柄
bdroot	获取根级系统名
simulink	打开 Simulink 模块库

2.1.2 要求一：指定命令的参数

与许多软件中的命令一样，在本章中提到的Simulink命令均要求指定描述系统、模块或模块参数的特定变量。读者必须了解这些信息，附录给出了所有模型和模块的参数表，请查阅。

2.1.3 要求二：指定执行对象的路径

本章中所提到的许多命令要求必须有确定的Simulink系统或模块对象，这通过指定它们的路径(名)来确定：

(1) 确认一个系统：指定系统名(不必带.mdl扩展名)。

```
system
```

(2) 确认一个子系统：按层次顺序指定，包括从系统名到目标子系统名，并用"/"分隔。

```
system/subsystem 1 /.../subsystem
```

(3) 确认一个模块：指定包含该模块的系统的路径和目标模块名。

```
system/subsystem 1 /.../subsystem/block
```

2.1.4 有关说明

(1) 如果命令中涉及到指定的系统(模型)，必须在该命令之前用open_system命令将模型打开。

(2) 一般而言，可直接运用命令进行相关操作。例如下面的命令将获得SineInte系统中名为Sine Wave模块的Sample time参数值。

```
>>open_system('SineInte')
>>get_param('SineInte/Sine Wave','Sample time')
ans =
0
```

(3)若模块名中包括一个换行符或回车，则须将模块名(所有路径)指定为一个字符串矢量，并用sprintf('\n')作为换行符。例如，下面的命令先将换行符赋值给cr，然后获取Signal Generator模块的Amplitude(幅度)参数值。

```
>>cr = sprintf('\n');
>>get_param(['untitled/Signal',cr,'Generator'],'Amplitude')
ans =
1
```

(4) 若模块名包括一个斜线号(/)，则当指定模块名时应保留。例如，下面的命令将获取mymodel系统中命名为Signal/Noise模块的Location参数值。

```
>>get_param('mymodel/Signal//Noise','Location')
```

当然，最好还是尽量避免出现这种情况为妙。

2.2 用命令建模实例

本节通过一个例子，说明用命令的方式建模的基本步骤。这里仍然以上章中的"对一个正

弦波进行积分运算的模型"为例。

注意:命令均在 MATLAB 命令窗口内输入,并且系统对象必须是处于打开状态。

步骤 1 建立新系统

这一步骤必须先行完成。

```
>>new_system('SineInte')          % 建立名为 SineInte 的系统,打开但不显示窗口
>>open_system('simulink3')        % 打开 Simulink 库窗口
```

步骤 2 复制模块

我们知道,模型中包括 Sine Wave,Scope,Integrator 和 Mux 等 4 个模块,所以需要在系统(模型)中建立这些模块,下面的命令从 Simulink 库中的内置模块(标准模块)复制到新建模型内。

```
>>add_block('built - in/Sine Wave','SineInte/Sine Wave');
>>add_block('built - in/Scope','SineInte/Scope');
>>add_block('built - in/Integrator','SineInte/Integrator');
>>add_block('built - in/Mux','SineInte/Mux')
```

上述命令执行后,模型内容应该是如图 2.1 所示,为了清楚,这里已将模块分开放置。当然也可以用后面介绍的 Set_system 命令设置模块的位置和大小。

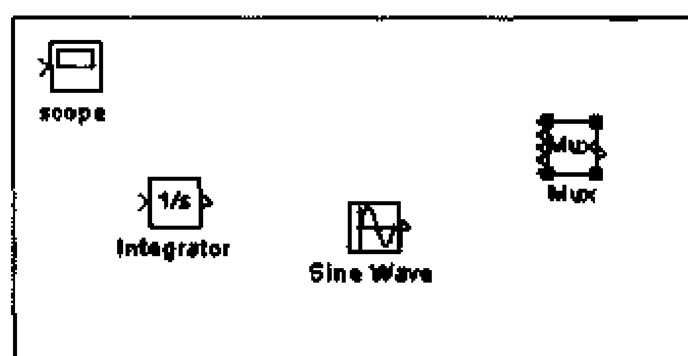


图 2.1 模型视图(1)

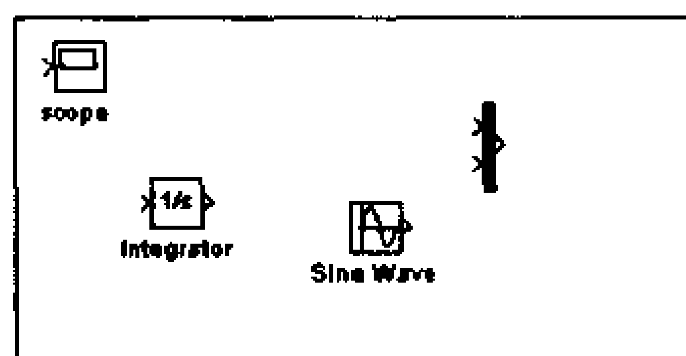


图 2.2 模型视图(2)

如果我们观察,就会发现,模块 Mux 的图标与以前的不同,但这无关紧要,读者可以通过双击该模块,打开对话框,将 Number of inputs 栏内的数字改为 2,并在 Display option 下拉菜单中选择 bar 后,单击 OK 按钮,然后将模块的外形进行调整即可。

步骤 3 连接

下面我们用连线将模块连接起来。为此,重新调整模块的相对位置,如图 2.3 所示。

下面的命令把 Sine Wave 输出和 Mux 的第一个输入端口连接起来,如图 2.4 所示。

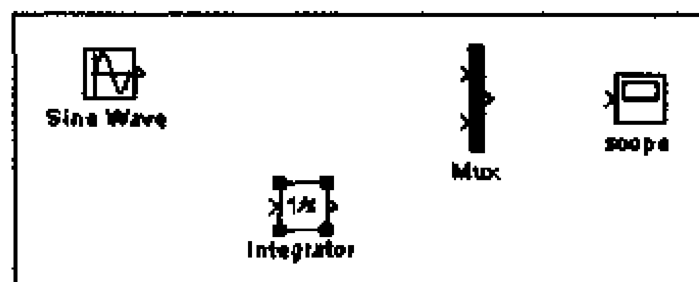


图 2.3 连接模块(1)

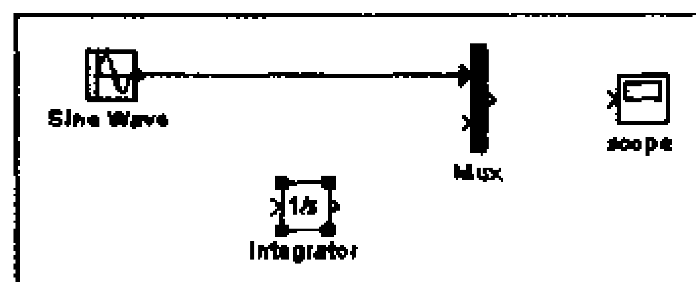


图 2.4 连接模块(2)

```
>>add_line('sineinte','Sine Wave/1','Mux/1')
```

下面的命令按定义连线位置的方法把 Sine Wave 模块的输出连线与 Integrator 的输入端

口连接起来，如图 2.5 所示。

```
>>add_line('sineinte',[80 30; 80 85; 120 85])
```

最后的模型如图 2.6 所示。

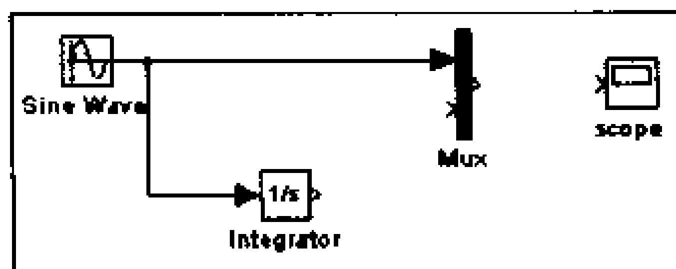


图 2.5 连接模块(3)

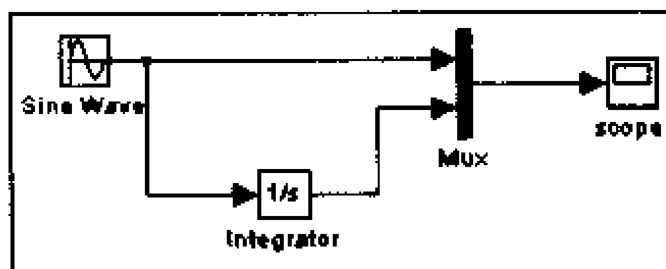


图 2.6 最后的模型(4)

下面逐一介绍所有命令，为了方便查阅，在命令的具体描述中，将按照命令的英文字母排列顺序进行。

2.3 add_block 命令

1. 功能

给已指定的系统(模型)添加一个模块。

2. 命令格式及描述

格式一：add_block('src', 'dest')

将完全路径名为"src"的模块复制为一个名为"dest"的新模块。新模块的模块参数与源模块完全相同。名'built-in'可以作为 Simulink 模块库中的非封装模块的源系统名。

格式二：add_block('src', 'dest', 'parameter1', value1, ...)

在复制新模块的同时，将指定的参数赋予指定的值。

3. 举例

(1) 如下命令将 Simulink 系统的 Sinks 子系统中的 Scope 模块复制为 engine 系统的 timing 子系统中的一个名为 Scope1 的模块，这里假设系统 engine 及其子系统 timing 已经打开，后同。

```
>>add_block('simulink3/Sinks/Scope', 'engine/timing/Scope1')
```

(2) 如下命令在 J-10A 系统中建构一个名为 controller 的新(子)系统。

```
>>add_block('built-in/SubSystem', 'J-10A/controller')
```

(3) 如下命令将内置模块 Gain 复制为系统 mymodel 中名为 Volume 的模块，并且将 Gain 参数赋值为 4。

```
>>add_block('built-in/Gain', 'mymodel/Volume', 'Gain', '4')
```

4. 相关命令

delete_block, set_param 命令。

2.4 add_line 命令

1. 功能

给指定 Simulink 系统添加一条连线，并返回一个新连线的句柄，按直接连线和分支线有两种实现方法。

(1) 利用连线连接的模块端口命名。

(2) 指定定义线段点的位置。

2. 命令格式及描述

格式一：h = add_line('sys', 'oport', 'iport')

add_line('sys', 'oport', 'iport')在指定模块输出端口'oport'与指定模块输入端口'iport'之间添加一连线。'oport'和'iport'是由指定模块名和一个端口标识符组成，格式为'block/port'。大多数模块端口是从上到下或从左到右编号标识的，如'Gain/1'或'Sum/2'。Enable(使能)、Trigger(触发)以及 State(状态)端口是通过端口名标识的，如'subsystem_name/Enable', 'subsystem_name/ Trigger'，或'Integrator/State'。

格式二：h = add_line('sys', points)

add_line(system, points)给一个系统添加一条分支连线。数组 points 每一行指定在线段上某一点的 x 和 y 坐标。原点在窗口的左上角。信号则从在第一行定义的点流向在最后一行定义的点。若新连线的起点靠近某一个已有的模块或连线，则它们就自动连接起来。同样，若连线的末端靠近一个已有的输入，则也自动连接。

3. 举例

下面的命令给 SineInte 系统添加一个连线，将 Sine Wave 模块的输出与 Mux 模块的第一个输入端口连通，如图 2.7 所示。

```
>>add_line('sineinte','Sine Wave/1','Mux/1')
```

下面的命令给 SineInte 系统添加一个连线，从(80,30)延伸到(80,85)，再到(120,85)。

```
>>add_line('sineinte',[80 30; 80 85; 120 85])
```

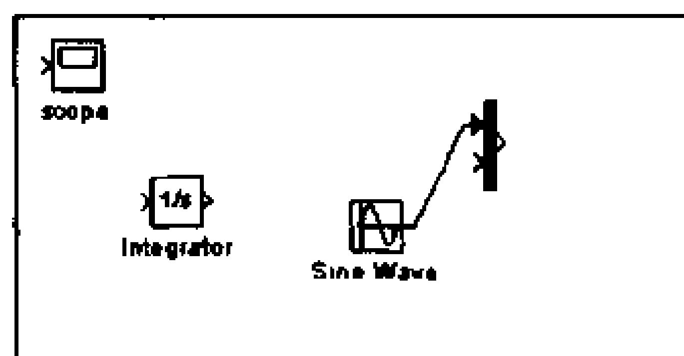


图 2.7 例图

4. 相关命令

delete_line 命令。

2.5 bdclose 命令

1. 功能

无条件关闭某一个或所有 Simulink 系统窗口。

2. 命令格式及描述

格式一：bdclose

此格式的命令将强制关闭当前系统窗口，并失去所有自最后一次保存以来对系统的修改。

格式二：bdclose('sys')

关闭指定的系统窗口。

格式三：bdclose('all')

关闭所有系统窗口。

3. 举例

下面命令关闭 SineInte 系统：

```
>>bdclose('SineInte')
```

4. 相关命令

close_system，new_system，open_system，save_system 命令。

2.6 bdroot 命令

1. 功能

返回最高级 Simulink 系统名。

2. 命令格式及描述

格式一：bdroot

无变量的 bdroot 命令返回最高级系统名。

格式二：bdroot('obj')

其中'obj'是一个系统或模块路径名，该命令返回包含指定目标名的最高级系统名。

3. 举例

我们以 SineInte1 系统为例。SineInte1 是在 SineInte 的基础上重新编辑而成，系统下包含 Sine Wave 和 Integrator 模块以及一个含有 Mux 和 Scope 模块的子系统 M+S。下面的命令返回含有当前模块的最高级系统名。

```
>>bdroot('SineInte1/M + S/Scope')
ans =
SineInte1
```

4. 相关命令

find_system，gcb 命令。

2.7 close_system 命令

1. 功能

关闭一个 Simulink 系统窗口或一个模块对话框。

2. 命令格式及描述

格式一:close_system

无变量 close_system 命令关闭当前系统和子系统窗口。若当前系统为最高级(根)系统,且经过修改,则 close_system 会给出用户在关闭系统之前是否保存对系统的修改等提示信息。当前系统的概念将在 gcs 命令中再做说明。

格式二:close_system('sys')

关闭指定系统或子系统窗口。

格式三:close_system('sys', saveflag)

关闭指定的最高级系统窗口,并将其从内存中清除。

若 saveflag 为数字 0,则系统不保存;若 saveflag 为数字 1,则系统以当前名保存。

格式四:close_system('sys', 'newname')

将指定的最高级系统保存在新指定的模型中,然后关闭系统。

格式五:close_system('blk')

式中'blk'是一个详尽的模块路径名,关闭与指定模块相关的对话框,或当模块的 CloseFcn 恢复参数已经定义过后调用该参数。其他任何变量将被忽略。

3. 举例

下面的命令关闭当前系统:

```
>>close_system
```

下面的命令将仅仅关闭子系统 M+S:

```
>>close_system('sineintel/M + S')
```

下面的命令关闭整个系统:

```
>>close_system('sineintel')
```

下面的命令以当前名保存 engine 系统,然后关闭:

```
>>close_system('engine', 1)
```

下面的命令以 sineinte2 名保存当前 SineIntel 系统,若 SineIntel 修改过,但未保存,则在关闭的同时,不保存修改。

```
>>close_system('sineintel','sineinte2')
```

下面的命令关闭 engine 系统中 Combustion 子系统的 Unit Delay 模块对话框:

```
>>close_system('engine/Combustion/Unit Delay')
```

4. 相关命令

bdclose,gcs,new_system,open_system,save_system 命令。

2.8　delete_block 命令

1. 功能

将一个模块从一个 Simulink 系统中删除。

2. 命令格式及描述

delete_block('blk')

式中'blk'是一个详尽的模块路径名,命令将一个模块从一个系统中删除。

3. 举例

下面的命令将 Scope 模块从 SineIntel 系统中删除:

```
>>delete_block('sineinte2/Scope')
```

4. 相关模块

add_block 命令。

2.9 delete_line 模块

1. 功能

删除一个 Simulink 系统中的一条连线。

2. 命令格式及描述

格式一:delete_line('sys', 'oport', 'iport')

delete_line('sys', 'oport', 'iport')命令删除从指定模块的输出端口'oport'到指定模块输入端口'iport'之间的连线。'oport'和'iport' 字符串由一个模块名及端口标识符组成,以'block/port'形式表示。大多数模块端口通过对端口进行从上到下或从左到右编号以便标识,如'Gain/1'或'Sum/2'等。使能端口、触发端口及状态端口是以名称进行标识的,如'subsystem_name/Enable','subsystem_name/ Trigger','Integrator/State'等。

格式二:delete_line('sys', [x y])

删除系统中含有指定坐标点(x,y)的一条连线。

3. 举例

下面的命令将 SineInte2 系统中 Sine Wave 模块与 Mux 模块的第一输人之间的连线删除:

```
>>delete_line('sineinte2','Sine Wave/1','Mux/1')
```

4. 相关命令

add_line 命令。

2.10 find_system 命令

1. 功能

查找系统、模块、连线以及注解。

2. 命令格式及描述

格式:find_system(sys, 'constraint', cv, 'p1', v1, 'p2', v2,...)

该命令根据 constraint 指定的约束条件搜寻 sys 指定的系统或子系统,并返回一个目标句柄或路径,该对象具有参数值 v1,v2 等。sys 可以是一个路径名(或路径名的单元数组)、一个句柄(或句柄矢量)或省略。若 sys 是一个路径名或路径名的单元数组,find_system 命令将返回搜寻到的目标路径名的单元数组;若 sys 是一个句柄或句柄矢量,find_system 命令在所

搜寻到的目标上返回一个句柄矢量；若 sys 省略，find_system 将搜索所有打开的系统。

参数名忽视空格，但数值字符串容纳空格。所有从对话框输入的参数都可以是字符串值。参看附录给出的模型和模块参数表。

用户可以指定如表 2.2 所示搜索约束条件之一：

表 2.2　搜索约束条件

名 称	数据类型	描 述
'SearchDepth'	标量	限制搜索深度，按指定级别：0 表示搜索打开的系统；1 表示搜索最高级系统的模块和子系统；2 表示搜索最高级系统及其子系统。默认值为所有级
'LookUnderMasks'	'on' | 'off'	若为'on'，搜索延伸至封装系统内。默认值为'off'
'FollowLinks'	'on' | 'off'	若为'on'，跟随链接进入库模块搜索。默认值为'off'
'FindAll'	'on' | 'off'	若为'on'，搜索扩展到系统内连线和注解。默认值为'off'

若'constraint'省略，find_system 将采用默认约束条件值。

3. 举例

下面的命令返回一个包含所有打开系统和模块名的单元数组：

```
>>find_system
ans =
        'SineIntel'
        'SineIntel/Integrator'
        'SineIntel/M + S'
        'SineIntel/M + S/In1'
        'SineIntel/M + S/In2'
        'SineIntel/M + S/Mux'
        'SineIntel/M + S/Scope'
        'SineIntel/Sine Wave'
        'SineInte'
        'SineInte/Integrator'
        'SineInte/Mux'
        'SineInte/Scope'
        'SineInte/Sine Wave'
```

下面的命令返回所有打开的方框图名：

```
>>open_bd = find_system('Type', 'block_diagram')
open_bd =
        'sineinte2'
        'SineIntel'
        'SineInte'
```

下面的命令返回 sineinte2 系统的 M+S 子系统的 Scope 模块名：

```
>>find_system('sineinte2/M + S','SearchDepth',1,'BlockType','Scope')
ans =
        'sineinte2/M + S/Scope'
```

下面这些命令返回所有在 vdp 系统中，并且 Gain 参数值为 1 的 Gain 模块名：

```
>>open_system('vdp')
>>gb = find_system('vdp', 'BlockType', 'Gain')
>>find_system(gb, 'Gain', '1')
gb =
'vdp/Mu'
ans =
'vdp/Mu'
```

这相当于下面的命令：

```
>>find_system('vdp', 'BlockType', 'Gain', 'Gain', '1')
```

下面这些命令获得 vdp 系统中所有连线和注解句柄：

```
>>sys = get_param('vdp', 'Handle');
>>l = find_system(sys, 'FindAll', 'on', 'type', 'line');
>>a = find_system(sys, 'FindAll', 'on', 'type', 'annotation');
```

4. 相关命令

get_param，set_param 命令。

2.11 gcb 命令

1. 功能

获取当前模块路径名。

2. 命令格式及描述

格式一：gcb

返回当前系统中当前模块的详尽路径名。

当前模块是指如下之一：

(1) 在编辑过程中，当前模块为最近点击过的模块；

(2) 在对包含 S-Function 模块的仿真过程中，当前模块为最近执行其相应 MATLAB 函数的 S Function 模块；

(3) 在恢复期间，当前模块为正在执行其恢复程序的模块；

(4) 在 MaskInitialization 字符串赋值期间，当前模块为正在封装赋值的模块。

格式二：gcb('sys')

返回指定系统中当前模块的路径名。

3. 举例

下面的命令返回最近选择的模块路径：

```
>>gcb
ans =
SineIntel/M + S/Scope
```

下面的命令获取当前模块 Scope 的 NumInputPorts 参数值：

```
>>get_param(gcb,'NumInputPorts')
ans =
1
```

4. 相关命令

gcbh 和 gcs 命令。

2.12 gcbh 命令

1. 功能

获取当前模块的句柄。

2. 命令格式及描述

格式：gcbh

返回当前系统中的当前模块的句柄。

用户可以使用本命令对子系统模块进行识别或编址。用于编辑模块组。

3. 举例

下面的命令返回最近选择的模块句柄：

```
>>gcbh
ans =
7.0001
```

4. 相关命令

gcb 命令。

2.13 gcs 命令

1. 功能

获取当前系统的路径名。

2. 命令格式及描述

格式：gcs

返回当前系统的详尽路径名。

当前系统指的是下列情况之一：

(1) 在编辑过程中，当前系统为最近点击过的系统或子系统；

(2) 在对包含 S-Function 模块的系统进行仿真过程中，当前系统为正在对 S-Function 模块进行赋值的系统或子系统；

(3) 在收回期间，当前系统为含有某个正在执行收回例程的模块的系统；

(4) 在 MaskInitialization 字符串赋值期间，当前系统为含有正在对封装的模块进行赋值的系统。

3. 举例

下面的命令返回含有最近选择的模块的系统名：

```
>>gcs
ans =
SineIntel
```

4. 相关命令

gcb 命令。

2.14 get_param 命令

1. 功能

获取系统和模块参数值。

2. 命令格式及描述

格式一：get_param('obj', 'parameter')

其中'obj'为某系统或模块的路径名，命令返回指定参数值。参数名忽略空格。

格式二：get_param({ objects }, 'parameter')

接受一个详尽路径区分符的单元数组，这使用户能得到所有在单元数组中指定的目标的共有参数值。

格式三：get_param(handle, 'parameter')

命令返回目标句柄的指定参数。

格式四：get_param('obj', 'ObjectParameters')

返回一个描述 obj 参数的结构。返回结构的每一栏对应一个详细的参数，并有参数名。例如，Name 栏对应于目标的 Name 参数。每个参数栏又包含三个栏：Name，Type 和 Attributes，它们分别指定参数的名(如“Gain”)，数据类型(如 string(字符串))以及属性(如 read-only(只读))。

格式五：get_param('obj', 'DialogParameters')

返回一个含有指定模块对话参数名的单元数组。

请参考附录列出的模型和模块的参数表。

3. 举例

下面的命令返回在 SineIntel 系统 M+S 子系统中 Scope 模块的 Ymin 参数值：

```
>>get_param('sineintel/M + S/Scope','Ymin')
ans =
 -5
```

下面的命令显示 mx+b 系统(当前系统)中所有模块的模块类型。

```
>>blks = find_system(gcs, 'Type', 'block');
>>listblks = get_param(blks, 'BlockType')
listblks =
'SubSystem'
'Inport'
'Constant'
'Gain'
'Sum'
'Outport'
```

下面的命令返回当前选定模块的名：

```
>>get_param(gcb, 'Name')
ans =
Integrator
```

下面的命令获得当前选定模块 Name 参数的属性：

```
>>p = get_param(gcb, 'ObjectParameters');
>>a = p.Name.Attributes
a =
    'read - write'    'always - save'
```

下面的命令获得一个 Sine Wave 模块的对话参数：

```
>>p = get_param('sineintel/Sine Wave', 'DialogParameters')
    p =
     Amplitude: [1x1 struct]
Frequency: [1x1 struct]
         Phase: [1x1 struct]
    SampleTime: [1x1 struct]
```

4. 相关命令

find_system,set_param 命令。

2.15　new_system 命令

1. 功能

创建一个新(空)Simulink 系统。

2. 命令格式及描述

格式:new_system('sys')

该命令创建一个新的系统,若'sys'指定了一个路径,则新系统将是路径指定系统的一个子系统。new_system 命令不打开系统窗口。欲知有关新系统默认参数值表,请查看附录。

3. 举例

下面的命令创建一个新的、名为'mysys'的系统:.

```
>>new_system('mysys')
```

下面的命令在 vdp 系统中创建一个新的、名为'mysys'的子系统。

```
>>new_system('vdp/mysys')
Warning: The use of new_system to create a subsystem will be obsoleted.
To add a subsystem block to 'sineintel' use
add_block('built-in/SubSystem','sineintel/mysys').
```

4. 相关命令

close_system, open_system, save_system 命令。

2.16 open_system 命令

1. 功能

打开一个 Simulink 系统窗口或一个模块对话框。

2. 命令格式及描述

格式一:open_system('sys')

打开指定系统或子系统窗口。

格式二:open_system('blk')

其中'blk'为详尽模块路径名,该命令打开指定模块的相关对话框。若模块的 OpenFcn 收回参数已经定义了,则程序赋值。

格式三:open_system('blk', 'force')

其中'blk'为详尽路径名或是一个封装系统,该命令在指定系统封装之下查找,相当于使用 Look Under Mask 菜单项。

3. 举例

下面的命令打开 controller 系统,并显示在默认位置:

```
>>open_system('controller')
```

下面的命令打开 controller 系统中 Gain 模块的模块对话框:

```
>>open_system('controller/Gain')
```

4. 相关命令

close_system, new_system, save_system 命令。

2.17 replace_block 命令

1. 功能

替换一个 Simulink 模型中的模块。

2. 命令格式及描述

格式一:replace_block('sys', 'blk1', 'blk2', 'noprompt')

该命令用'blk2'模块替换'sys'中所有模块或封装类型为'blk1'的模块。若'blk2'是一个Simulink 定制模块,只需模块名。若'noprompt'省略,Simulink 将显示一个对话框要求用户在替换之前选择匹配模块。指定'noprompt'自变量后对话框将不会显示。若用户指定了一个返回变量,被替换的模块的路径就会存入该变量中。

格式二:replace_block('sys', 'Parameter', 'value', 'blk', ...)

该命令用'blk'模块替换'sys'中所有指定参数为设定值的模块。用户可以指定任意数量的参数/值对。

注意:由于命令改变后难以撤消,所以最好先将系统保存。

3. 举例

下面的命令用 Integrator 模块替换 f14 系统中所有 Gain 模块,并将替换模块的路径存入 RepNames 中。Simulink 在替换之前在一个对话框中列出匹配的模块。

```
>>RepNames = replace_block('f14','Gain','Integrator')
```

下面的命令用 Integrator 模块替换 clutch 系统 Unlocked 子系统中所有 Gain 参数为'bv'的模块。Simulink 在替换之前在一个对话框中列出匹配的模块。

```
>>replace_block('clutch/Unlocked','Gain','bv','Integrator')
```

下面的命令用 Integrator 模块替换 f14 系统中所有 Gain 模块,但不显示对话框。

```
>>replace_block('f14','Gain','Integrator','noprompt')
```

4. 相关命令

find_system, set_param 命令。

2.18 save_system 命令

1. 功能

保存一个 Simulink 系统。

2. 命令格式及描述

格式一:save_system

保存当前系统。

格式二:save_system('sys')

将一个指定系统存入当前名的文件中,但系统必须是打开的。

格式三:save_system('sys', 'newname')

命令将指定的最高级系统存入一个指定名的新文件中,但系统必须是打开的。

3. 举例

下面的命令保存当前系统:

```
>>save_system
```

下面的命令保存 SineInte 系统:

```
>>save_system('SineInte')
```

下面的命令将 SineInte 系统存入名为'mysineinte'的文件中：

```
>>save_system('sineinte', 'mysineinte')
```

4. 相关命令

close_system，new_system，open_system 命令。

2.19 set_param 命令

1. 功能

设置 Simulink 系统和模块参数。

2. 命令格式及描述

格式：set_param('obj', 'parameter1', value1, 'parameter2', value2, ...)

'obj'为一个系统或模块的路径，该命令将指定参数设置为指定值。参数名忽略空格。值字符串容纳空格。从对话框输入的参数均为字符串数值。模型和模块参数列见附录。

用户可以在仿真期间改变工作空间的模块参数值，并且通过这些改变更新模块图。为此，先在命令窗口改变参数值，然后激活模型窗口，最后选中 Edit 菜单中的 Update Diagram 即可。

注意：大多数模块初始值必须指定为字符串。但 Position 和 UserData 参数除外。

3. 举例

下面的命令设置 vdp 系统的 Solver 和 StopTime 参数：

```
>>set_param('vdp', 'Solver', 'ode15s', 'StopTime', '3000')
```

下面的命令将 SineInte 系统的 Sine Wave 模块的 Sample time 参数设置为 0.01(恒定)：

```
>>set_param('sineinte/Sine Wave', 'Sample time', '0.01')
```

下面的命令设置 SineInte 系统中 Integrator 模块的位置：

```
>>set_param('sineinte/Integrator', 'Position', [120 100 150 130])
```

下面的命令设置 mymodel 系统中 Zero-Pole 模块的 Zeros 和 Poles 参数：

```
>>set_param('mymodel/Zero-Pole','Zeros','[2 4]','Poles','[1 2 3]')
```

下面的命令设置一个封装子系统中某模块的 Gain 参数，变量 k 与 Gain 参数有关：

```
>>set_param('mymodel/Subsystem', 'k', '10')
```

下面的命令设置 mymodel 系统中名为 Compute 模块的 OpenFcn 调回参数。当用户双击 Compute 模块时，执行'my_open_fcn'函数。

```
>>set_param('mymodel/Compute', 'OpenFcn', 'my_open_fcn')
```

4. 相关命令

get_param，find_system 命令。

2.20 simulink 命令

1. 功能

打开 Simulink 模块库。

2. 命令格式及描述

格式：simulink

本命令只适用于 WINDOWS 操作系统。命令打开 Simulink 模块库浏览器窗口。若 Simulink 模块库已经打开，发布此命令将激活 Simulink 窗口。在 SIMULINK4.0 版本中，该命令在同一个窗口上打开库浏览器和库窗口。

请注意本命令与下面的命令之间的不同。

2.21 simulink3 命令

1. 功能

打开 Simulink 模块库。

2. 命令格式及描述

格式：simulink3

命令打开 Simulink 模块库窗口。若 Simulink 模块库已经打开，发布此命令将激活 Simulink 窗口。

第3章 Simulink 仿真模型编辑器

通过前两章的学习，读者不仅已经了解了 Simulink 的特点和利用交互式或命令的方法建立简单仿真模型的步骤。在此基础上，本章将主要介绍关于仿真模型的基本编辑功能、方法以及一些细节描述，使读者特别是初学者进一步学习、掌握如何解决在模型的构建和仿真中必须面对的问题，另外包括许多提示信息的说明。

本章要点：Simulink 仿真模型的编辑与模型文件的操作方法。对初学者来说，这是运用 Simulink 进行建模、仿真的基础。

学习目的：进一步掌握 Simulink 仿真模型的构建方法。

3.1 Simulink 模型编辑概述

为了完整起见，我们这里仍然从进入 Simulink 开始。

3.1.1 进入 Simulink

启动 Simulink，有两条路径：

(1) 单击 MATLAB 工具栏上的 Simulink 图标。

(2) 在 MATLAB 窗口键入命令：

```
>>simulink
```

在 Windows 平台下，启动 Simulink 后就出现了 Simulink 库浏览器（如图 3.1 所示）。

库浏览器显示了安装在用户系统中的 Simulink 模块库的树状结构图。用户可以通过从库浏览器向模型窗口拷贝模块来构建模型。



图 3.1 Simulink 库浏览器窗

在 UNIX 平台下，启动 Simulink 后，出现的是如图 3.2 所示的库窗口。正如在第 1 章所述，Simulink 库窗口显示的是代表模块库的图标。用户可以通过从模块库向模型窗口拷贝模

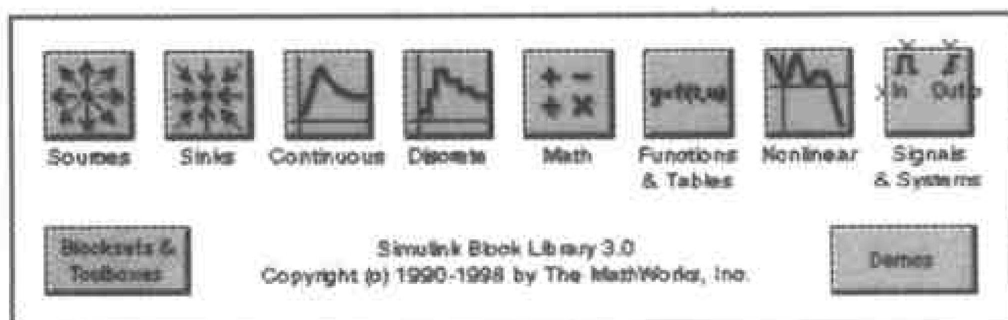


图 3.2 UNIX 平台下 Simulink 模块库窗口

块来构建模型。在 Windows 下，可以在库浏览器窗口中右键单击 Simulink 结点来显示 Simulink 库窗口。

3.1.2 构建新模型

在 Windows 下构建新模型，在库浏览器工具栏上单击 New 按钮，或者在库窗口的 File 菜单中选择 New，进而选择 Model。第 1 章和第 2 章介绍了构建一个简单模型的方法。如需要进一步了解更多的信息，可参看本章后面的内容。

3.1.3 编辑已有的模型

可以利用下列两种方法之一来编辑已有的模型框图：

(1) 选择库浏览器工具栏上的 Open 按钮(仅在 Windows 下)或者在 Simulink 库窗口的 File 菜单中选择 Open 命令，然后选择或输入需要编辑的模型文件名。

(2) 通过在 MATLAB 命令窗口中输入命令：

① 输入模型名(不带扩展名.mdl.)，如

```
>>SineInte
```

编者提示：模型必须在当前目录或路径下，系统默认路径/work/。输入字符要求严格区分。

② 用 open_system 命令，如：

```
>>open_system('SineInte')
```

3.1.4 Simulink 命令的输入

如任何一种程序一样，运行 Simulink 或对模型进行操作等控制，都是通过输入命令来完成的。输入命令的方法无非包括以下几种：

1. 通过 Simulink 菜单栏输入命令

Simulink 菜单栏出现在接近每个模型窗口顶部的位置。菜单命令仅适用于窗口内部的对象。

2. 使用下拉菜单输入命令(仅适用于 Windows)

当用户在一个模型窗口或模块库窗口上单击鼠标右键时，Simulink 会显示出一个下拉菜单，菜单的内容(命令)取决于模块是否被选中。如果有模块被选中，则菜单就显示那些仅适用于选中模块的命令；若没有模块被选中，菜单显示的就是那些对所有模型或模型库都适用的公共命令。

3. 使用 Simulink 工具栏输入命令(仅适用于 Windows)

默认时，模型窗口总是显示工具栏。如果没有显示而又需要这个工具栏，可通过选择 Simulink 的 View 菜单下的 Toolbar 项就可以了，如图 3.3 所示。

工具栏包含了经常使用的 Simulink 命令，比如打开、运行和关闭模型等。单击相应的命令按钮，就可完成这些命令。例如，要打开一个 Simulink 模型，只须单击带有打开文件夹图标的按钮即可。将鼠标箭头放在按钮上，就会出现一个描述该按钮作用的小窗口(称为工具提示窗口)，这样就可以确定该按钮到底执行哪条命令。当鼠标箭头盘旋在工具栏中的任意一个按钮上方时，都会显示一个工具提示。可以通过 Simulink 的 View 菜单中的 Toolbar 选项来隐

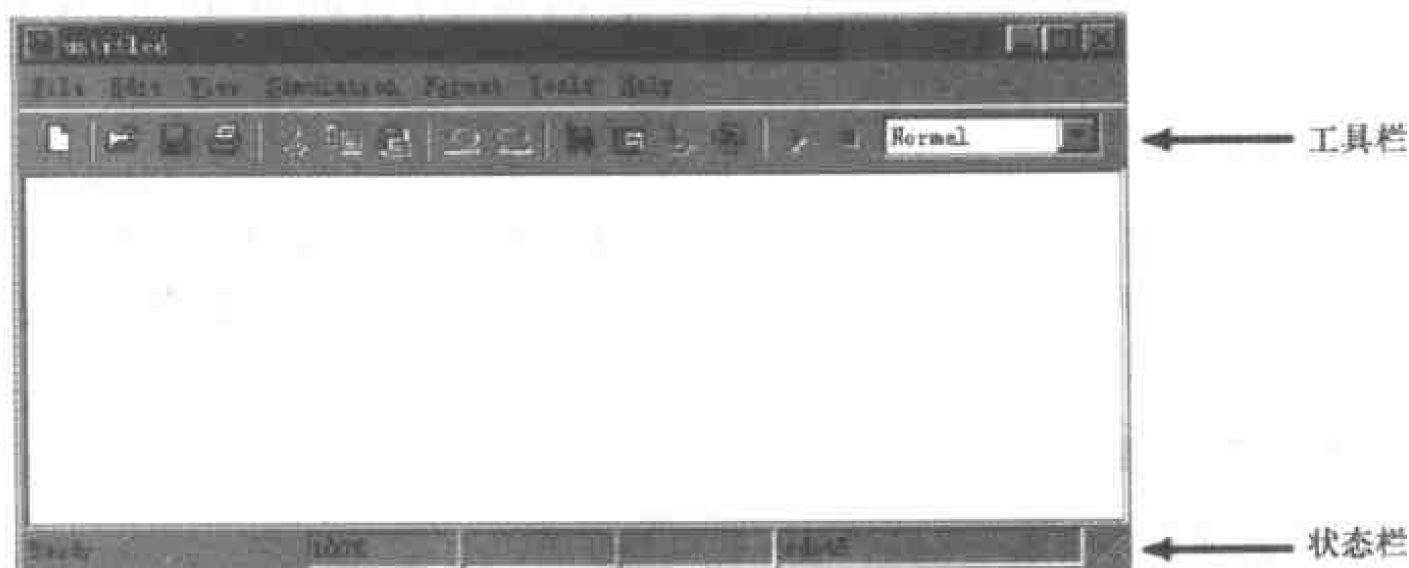


图 3.3 Simulink 工具栏和状态栏位置

藏工具栏。

4. 使用 MATLAB 窗口输入命令

当用户需要运行一个仿真并进行结果分析时，可以在 MATLAB 命令窗口中输入 MATLAB 命令。运行仿真的问题将在第 5 章讲述。

5. 撤消命令

通过在 Edit 菜单中选择 Undo 命令，可以撤消多达 101 次的连续操作。可以撤消的操作项目包括：

(1) 添加或删除模块；

(2) 添加或删除线；

(3) 添加或删除模型注释；

(4) 编辑模块名。

也可以在 Edit 菜单上选择 Redo 项来取消 Undo 命令执行的结果。

3.1.5 Simulink 窗口

Simulink 通过各自不同的窗口分别显示模块库浏览器、模块库、模型以及图形(示波器)仿真输出。

编者提示：这些窗口不是 MATLAB 图形窗口，不能使用句柄图形(Handle Graphics)命令进行操作。

Simulink 的窗口大小是按照最通用的屏幕分辨率设定的。如果用户的监视器分辨率很高或者很低，窗口就有可能太小或者太大。在调整窗口的大小后，系统会将参数保存下来以保持新窗口的尺寸。

3.1.6 状态栏

Windows 版本的 Simulink 在每个模型和库窗口的底部都会显示一个状态栏，见图 3.3。

在仿真运行期间，状态栏可以显示该仿真的运行状况，包括当前仿真时间及当前 Solver (称仿真器)名。用户可以通过在 Simulink 的 View 菜单中选择或不选择 Status Bar 项，来显示或者隐藏状态栏。

3.1.7　放大和缩小模型图

Simulink 可以放大或缩小在当前 Simulink 窗口中的模型图。具体操作如下：

(1) 在 View 菜单中选择 Zoom In(或键入 r)以放大视图。

(2) 在 View 菜单中选择 Zoom Out(或键入 v)以缩小视图。

(3) 在 View 菜单中选择 Fit System to View 以调整视图到适合观察的尺寸。

(4) 在 View 菜单中选择 Normal 以调整方框图到实际尺寸。

在默认情况下，当打开模型图时，无论是在模型浏览器的内容面板，还是在一个单独的窗口，Simulink 都会将模型图调整到适合观察的尺寸。如果改变了模型图的尺寸设置，在关闭时，Simulink 会自动保存该设置，并在下次打开这张图时恢复该设置。如果要还原默认设置，可在下次打开后选择 View 菜单下的 Fit System to View。

3.2　选择对象

许多编辑模型的操作，比如复制一个模块或删除一条线，都需要首先选择一个或多个模块和线(通称为对象)。

3.2.1　选择一个对象

要选择一个对象，只需在它上面点击即可。选中后，在选中模块的拐角或选中线的端点附近，会出现小的黑色正方形(句柄)。例如，图 3.4 就分别表示出选中的正弦波(Sine Wave)模块和选中的线：

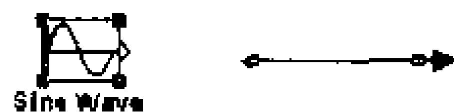


图 3.4　被选中后的模块和线示意图

当通过点击选择一个对象时，其他已经选定的所有对象都将撤消选定。

3.2.2　选择多个对象

要选择多个对象，可以通过一次选中一个的方式，或者采用方形选择框来选中彼此邻近的对象的方式，也可以通过选中整个模型的方式。

1. 一次选一个对象

通过一次选中一个的方式来选择多个对象，就是在按下 shift 键的同时单击各个需要选择的对象。要撤消一个选定的对象，可以在按下 shift 键的同时，再次单击该对象。

2. 采用方形选择框

要在一个窗口的同一区域内选择多个对象，简单的方法就是画一个方形框来框住所要选择的对象，方法是：

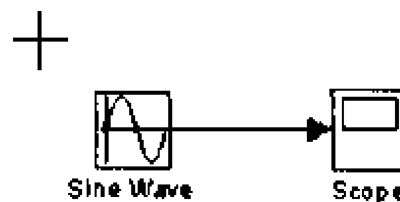


图 3.5　选择多个对象的步骤 1

(1) 将鼠标箭头放在方形框的一角以确定其开始角，然后按下鼠标键不放，参见图 3.5，注意光标的形状。

(2) 将鼠标箭头拖到方形框的对角，这时会出现一个虚线框，虚线框框住了需要选中的模块和线，参见图 3.6。

(3) 放开鼠标键，则所有被虚线框框住的模块和线(即使它们只有一部分被框住)就被选中了，如图3.7所示。

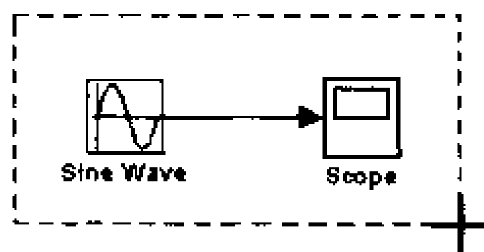


图3.6 选择多个对象的步骤2

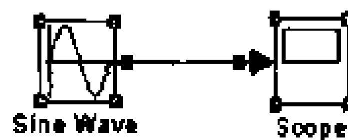


图3.7 选择多个对象的步骤3

3. 选择整个模型

要选择当前窗口的所有对象，只需在Edit菜单中选择Select All项。但不能通过这种选择模块和线的方式去创建子系统。更多的信息请参见"创建子系统"一节。

3.3 模块及其编辑

模块是构建Simulink模型的基本单元。可以将各种模块和线进行适当的连接，对任意动态系统进行仿真。本节将重点讨论如何使用模块来构建动态系统的模型。

3.3.1 模块信息提示

在Windows操作系统下，当把鼠标指针放在模型图标上时，Simulink会在一个弹出式的窗口内显示模块信息。要取消这个特性或者控制数据提示的信息内容，可选择Simulink View菜单中的Block Data Tips项。

3.3.2 虚拟模块

在构建模型时，应该注意Simulink的模块有两种基本类型：非虚拟模块和虚拟模块。非虚拟模块在系统仿真中是主动的。如果添加或删除一个非虚拟模块，模型的性质就改变了。相反，虚拟模块并不是主动的，它们仅仅是协助图形化组织模型。有些Simulink模块在一定环境下是虚拟的，而在另外环境下就可能是非虚拟的，这样的模块称作条件虚拟模块。表3.1列出了Simulink的所有虚拟和条件虚拟模块。

表3.1 虚拟模块

模块名	成为虚拟模块的条件
Bus Selector	总为虚拟
Data Store Memory	总为虚拟
Demux	总为虚拟
Enable Port	总为虚拟
From	总为虚拟
Goto	总为虚拟

续表 3.1

模块名	成为虚拟模块的条件
Goto Tag Visibility	总为虚拟
Ground	总为虚拟
Inport	一般为虚拟，除非该模块处于条件执行子系统中且与一个 Outport 模块直接相连
Mux	总为虚拟
Outport	当该模块处于一子系统模型(条件或无条件)中且不在 Simulink 根(顶层)窗口时，为虚拟
Selector	总为虚拟
Subsystem	当该模块不是条件执行时，为虚拟
Terminator	总为虚拟
Test Point	总为虚拟
Trigger Port	当不使用输出端口时，为虚拟

3.3.3 模块的复制和移动

1. 在模型内移动或复制模块

(1) 模块定位网格。Simulink 利用五倍数像素网格(默认时为不可见)来简化模块的整列。模型中的所有模块都被锁定在一条网格线上。

在 MATLAB 窗口内键入下面的命令，可以使网格在模型窗口中显示出来：

```
>>set_param('sineinte','showgrid','on') % 默认时，网格为 20 像素
```

键入如下命令可将网格间隔改为 10 像素：

```
>>set_param ( '<模型名>','gridspacing',10)
```

对于上面的两条命令，都可以先选中模型，然后键入 gcs，而不必输入<模型名>。

(2) 模块的移动。要在一个模型窗口里把单个模块从一个地方移动到另一个地方，只需将该模块拖到新位置，然后参照定位网格，按向上、向下、向左或向右等方向键略微地移动模块来调整模块的位置。Simulink 系统会同步而且自动调整连接到该模块的连线。

要同时移动多个模块以及连线，可以按照下列步骤操作：

① 选择模块和连线。有关如何选择多个模块的信息，请参看“选择多个对象”一节。

② 将所选定对象拖到新位置，然后释放鼠标键。

(3) 在同一个模型中复制模块。可以在同一个模型中复制模块，其方法是在按下 Ctrl 键的同时用鼠标点击模块，并将模块拖到新位置。也可以使用鼠标右键拖动模块来实现上述操作。复制的模块与原模块具有相同的参数值。Simulink 都会自动给新复制的模块指定一个名(模块名＋序号)。如复制模型中的 Sine Wave 后，新的模块名为 Sine Wave1。

2. 在两个模型间进行复制或移动

在构建模型时，常常需要从 Simulink 模块库或其他库、模型中将模块复制到用户自己的模型窗口。

方法一：

(1) 打开需要的模块库或模型窗口。

(2) 将需要复制的源模块拖动到目标模型窗口。拖动模块时，将光标放在该模型图标之上并按下鼠标不放，然后将光标拖到目标窗口内合适位置以后再释放鼠标键。

方法二：

还可以从 Simulink 库浏览器内将模块移动到模型窗口(可参阅"浏览模块库"一节)。

编者提示：在从 Simulink 模块库向模型中复制 Sum，Mux，Demux 和 Bus Selector 等模块时，Simulink 将自动隐藏它们的名字，实际上，这些模块的形状已经清楚地说明了它们各自的功能。

方法三：

利用 Edit 菜单中的 Copy 和 Paste 命令来复制模块。

(1) 选中需要复制的源模块。

(2) 在 Edit 菜单中选择 Copy(复制)项。

(3) 激活目标模型窗口。

(4) 在 Edit 菜单中选择 Paste(粘贴)项。

无论哪种方法，Simulink 都会自动给每个复制的模块指定一个名。如果它是模型中该类模块的第一个，它的名就与它在源窗口中的名相同。比如，如果从 Math 库中向目标模型窗口复制 Gain 模块，那么在目标模型中的这个新模块的名仍是 Gain。但是如果目标模型中已经包含一个名为 Gain 的模块，Simulink 就会在该模块名的后面加一个序号(如 Gain1，Gain2)。用户也可以给模块重新命名，详见"模块名的处理"一节。当然，新的复制模块将继承源模块的所有参数值。

任意模块都可以利用 Copy，Cut 和 Paste 命令复制或移动到兼容的应用程序(如字处理程序)中去。只是这些命令只能复制模块的图形，而不包含它们的参数。同样，可以使用 Edit 菜单中的 Undo 命令撤消已添加的模块。

至于在两个模型间移动模块，与模型间模块的复制类似，只不过在选定源模块后，不选择 Copy 而是 Cut。

3.3.4　设定模块参数

用户能够通过 Simulink 的用户界面指定模块参数值。有些模块参数对所有模块都是通用的，可以用 Block Properties 对话框来设定这些参数。要显示这个对话框，应先选中需要改变参数的模块，然后在 Edit 菜单中选择 Block Properties 项。具体可见"模块属性对话框"一节。

其他模块参数是针对个别具体模块的，可以使用模块的特定模块参数对话框来设置这些参数。双击该模块即可打开对话框。可以接受也可以改变显示的参数值。还可以使用 set_param 命令来改变模块参数。

一些模块的对话框允许用户指定它的某些或所有参数的数据类型。第 7 章将给出各个模块的对话框并介绍模块参数。

3.3.5 模块的属性对话框

利用 Block Properties 对话框(图 3.8 所示)可以设定一些通用模块的参数。该对话框包含如下内容：

1. 描述(Description)

本模块的简短描述。

2. 优先级(Priority)

执行本模块的优先级与模型中的其他模块有关系。详情请看“指派模块优先级”一节。

3. 标记(Tag)

与本模块一起保存的一种普通文本。

4. 调用函数(Open function)

用户打开本模块时调用的 MATLAB(m－)函数。

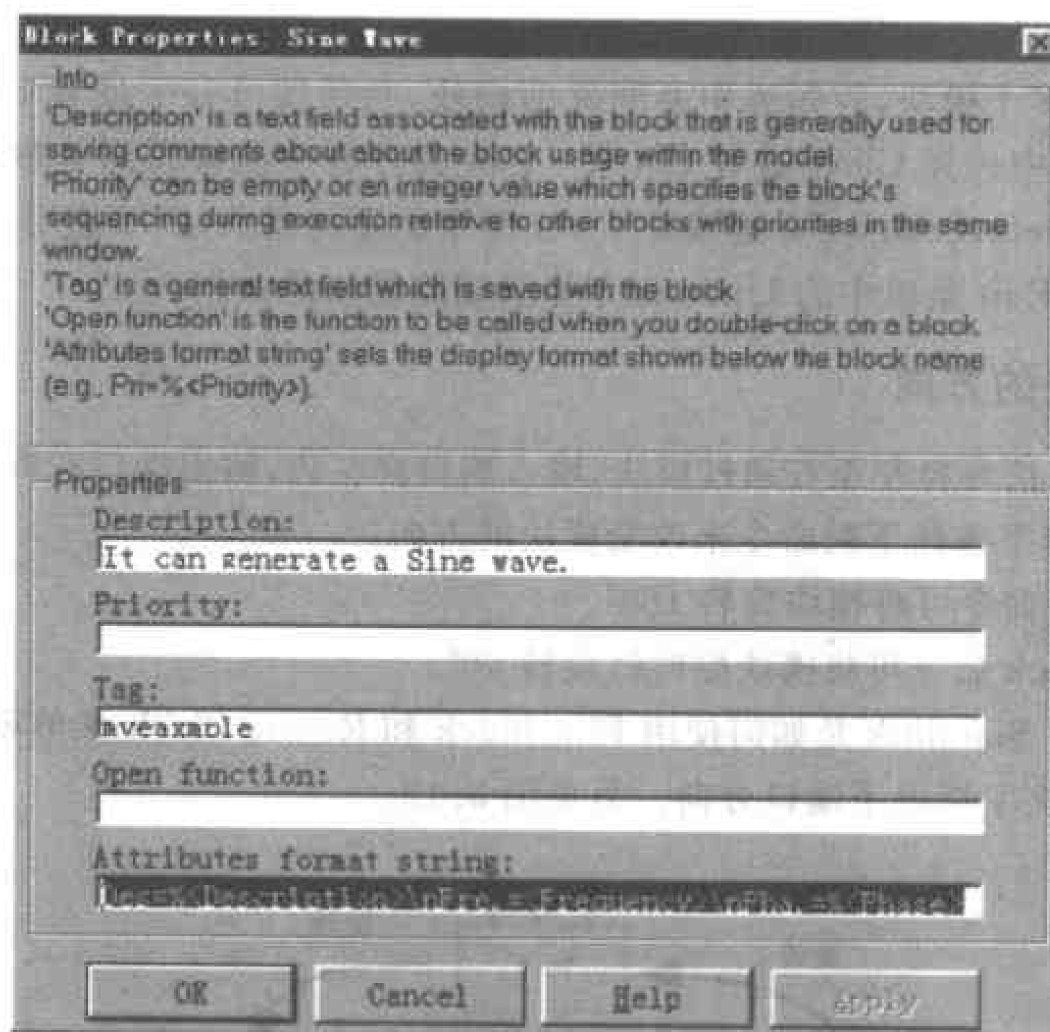


图 3.8 Sine Wave 属性对话框

5. 属性格式字符串(Attributes format string)

指定在该模块的图标下显示模块的参数或属性项。属性格式字符串可以是任意带有嵌入式参数名的文本字符串。嵌入式参数名以“%＜”开头，以“＞”结尾，例如%＜priority＞。Simulink 在模块的图标下显示属性格式字符串，以相应的参数值替换各个参数名。若有多个字符串，用换行符(\n)来分行显示各个参数。例如，在一个 Sine Wave 模块的属性格式字符串栏内输入：

```
Des = %<Description>\nFre. = %<Frequency>\nPha. = %<Phase>\nSTime = %
```

<Sametime>

经确定，则显示如图 3.9。

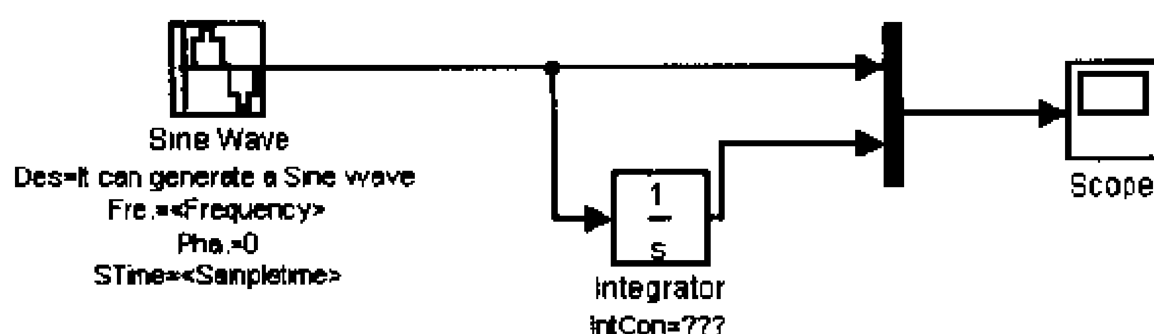


图 3.9　gain 模块显示的属性信息

如果参数值不是一个字符串或整数，则 Simulink 在参数值的位置处显示 N/S(不支持)。如果参数名无效，则显示"???"，如图 3.9 中 Integrator 模块下方所示。

3.3.6　删除模块

要删除一个或多个模块，首先选中要删除的模块，然后按 delete 或者 Backspace 键即可；也可以从 Edit 菜单中选择 Clear 或者 Cut 项。Cut 命令将该模块写到剪贴板上，可以用来粘贴到模型。使用 delete、Backspace 键或者 Clear 命令删除的模块，则不能用于以后的粘贴。

同样可以使用 Edit 菜单中的 Undo 命令撤消删除模块的操作。

3.3.7　改变模块的方向

在默认情况下，信号自左至右通过模块，输入端口在左边，输出端口在右边。若需要，可以通过在 Format 菜单下选择下列命令来改变模块的方向：

(1) Flip Block 命令可将模块旋转 180°。

(2) Rotate Block 命令可将模块顺时针旋转 90°。

图 3.10 显示了 Simulink 是如何使用 Flip Block 和 Rotate Block 菜单项改变模块端口的方向的。模块内的文字指出了端口方向。另见图 3.12。

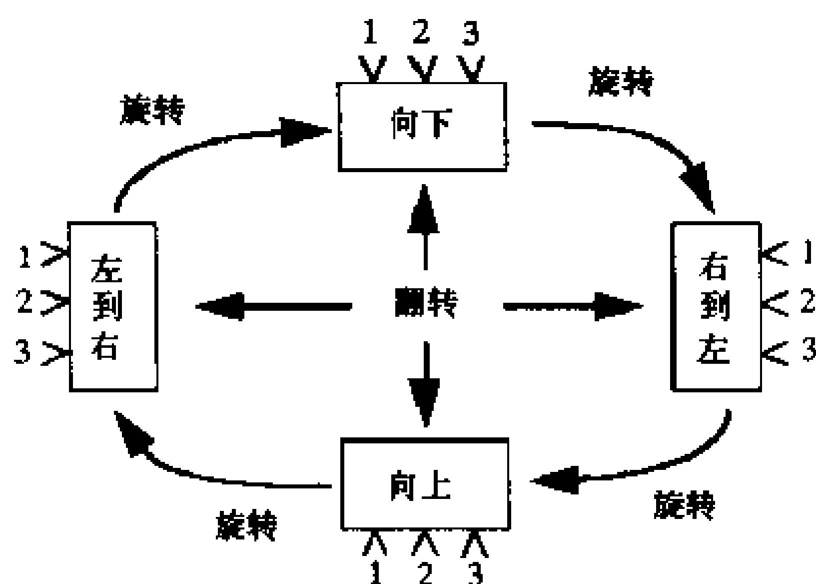


图 3.10　改变模块方向的顺序

3.3.8 调整模块大小

选择一个模块，然后按下鼠标，拖动模块四个选中句柄中的任意一个。这时会出现一个新的虚线框，表示新模块的大小。释放鼠标键后，模块大小就改变了。

例如，图 3.11 给出了一个要调整其大小的显示器模块。模块的右下句柄被选中并被拖动到光标位置。放开鼠标键时，模块的尺寸已经改变了。

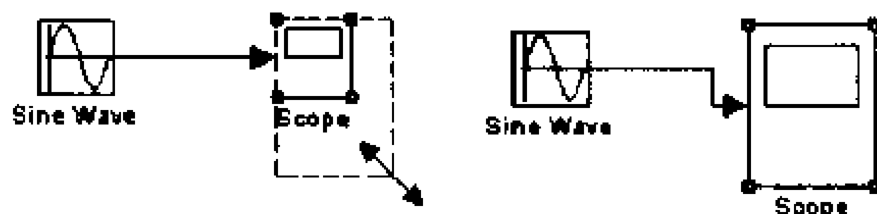


图 3.11 模块大小调整前后

3.3.9 编辑模块名

在一个模型里，所有的模块名必须各不相同并且至少包含一个字符。在默认情况下，当模块端口在两侧时，模块名出现在模块下面；当模块端口是上下时，模块名出现在模块的左边，如图 3.12 所示。



图 3.12 模块名位置

1. 改变模块名

可以采用下面几种方法之一来编辑模块名：

(1) 在 Windows 或者 UNIX 系统下，要替换模块名，只需单击该模块名，然后双击或拖动光标选中整个名字符串，再键入新名称即可。

(2) 要插入字符，则在需要插入的两个字符之间单击，然后插入字符。

(3) 要替换字符，则应拖动鼠标选中要替换的部分，然后键入新字符。

当在模型的其他地方单击鼠标或者执行其他操作时，新模块名就有可能被认可或者拒绝。如果输入的新模块名已经存在或没有字符，Simulink 就会显示出错信息。

要改变模块名的字体，应先选中该模块，然后在 Format 菜单下选择 Font 项，再从 Set Font 对话框中选择一种字体。此操作也同时改变了模型图标中的文本字体。

可以通过 Edit 菜单下的 Undo 命令撤消已做的编辑。

编者提示：如果改变库模块的名，则对该模块的所有链接都将无效。

2. 改变模块名的位置

改变一个选中的模块名的位置有下面两种途径：

(1) 将模块名拖动到模块的另一边。

(2) 选择 Format 菜单中的 Flip Name 项。这个命令可以将模块名的位置移到相反的一边。

参见 3.3.7 节“改变模块的方向”。

3. 改变模块名的显示状态

要改变一个模块名的显示状态，可以选择 Format 菜单中的一项：

(1) Hide Name 项将隐藏模块名。当选择隐藏时，Hide Name 变成 Show Name。

(2) Show Name 项将显示隐藏了的模块名。

3.3.10 在模型图标下显示模块参数

Simulink 可以在方框图中的模型图标下显示一个或多个模块参数。可以通过下面的方法指定要显示的参数：

(1) 在模块的 Block Properties 对话框的 Atrributes format string 栏内输入属性格式字符串(详见"模块属性对话框"一节)。

(2) 使用 set_param 命令，设置模块的属性格式字符串(参见第 2 章"set_param 命令")。

3.3.11 断开模块

按下 shift 键不放并将模块拖拉到新位置，可以将一个模块从它的连接线上断开。

3.3.12 矢量输入和输出

几乎所有的 Simulink 模块都能接受标量或矢量输入，产生标量或矢量输出，并且允许用户提供标量或矢量参数。在本书中，认为这些模块是矢量化(vectorized)的。

通过选择 Format 菜单中的 Wide Vector Lines 项，可以使模块间传输矢量的连线较传输标量的更粗一些，这样就能够确定模型中哪些连线传输的是矢量信号了。下一节将给出显示标量和矢量连线的示图。如果选择了 Wide Vector Lines 命令改变了模型，则必须在 Edit 菜单中选择 Update Diagram 项以更新显示(启动仿真同样也会更新方框图的显示)。

第 7 章将给出所有模块的输入、输出和参数特性。

3.3.13 输入和参数的标量扩展

所谓标量扩展(Scalar expansion)，就是把一个标量值转换成具有相同元素的矢量。Simulink 对大部分模块的输入变量和/或参数都可以进行标量扩展。在第 7 章中会给出各个模块能否对输入和参数进行标量扩展。

1. 输入的标量扩展

当使用有多个输入端口的模块(如 Sum 模块或 Relational Operator 模块)时，可以将矢量输入与标量输入混合起来使用。这时，这个标量输入便被广增为具有相同元素的矢量输出，其宽度与输入矢量的宽度相同(如果有多个输入模块是矢量，则它们的元素数必须相同)。

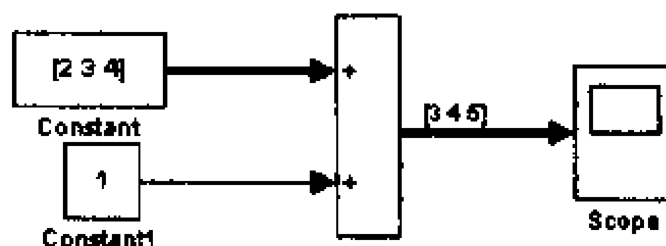


图 3.13 某模型组成

图 3.13 所示的这个模型就是将标量输入与矢量输入相加。来自模块 Constant1 的标量输入为了与来自模块 Constant 的矢量输入相匹配，就被广增为三个具有相同值元素的矢量[1 1 1]。

2. 参数的标量扩展

用户既可以将矢量模块的参数设定为矢量，也可以设定为标量。当设定为矢量参数时，每个参数元素与输入矢量中的相应元素都是相关联的；当设定为标量参数时，Simulink 则利用标量扩展将它们自动转换为具有适当维数的矢量（更一般地，在 Simulink 4.0 版本中，用 Dimentionazied 代替了 Vectorized，详见第 8 章）。

图 3.14 所示的例子表明一个标量参数（增益 Gain）被广增为具有相同数值元素的矢量，其宽度与该模块的输入矢量——一个三元素矢量——的宽度相同。

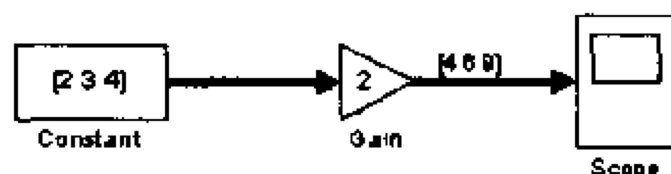


图 3.14 Gain 模块标量扩展

3.3.14 指派模块优先级

用户可以在模型中指派非虚拟模块的运算优先级。较高优先级的模块在较低优先级的模块之前运算，但这并不意味着指定优先级模块的运算一定会在那些没有指派优先级的模块之前进行。

可以用交互方式，也可以用编程（命令）方式指派模块的优先级。以编程方式指派优先级，应使用命令：

```
set_param ( b,'Priority','n')
```

其中 b 是模块路径，n 是任意有效整数（负数和 0 也是有效的优先值）。优先值越低，优先级越高，如 2 比 3 的优先级高。以交互方式指定模块优先级，只需在 Block Properties 对话框的 Priority 域输入优先值即可（参看“模块属性对话框”一节）。

3.3.15 使用阴影

选中模块，然后在 Format 菜单下选择 Show Drop Shadow 项，就可以给模块加上阴影。当选中的模块使用阴影时，该菜单项就变为 Hide Drop Shadow（隐藏阴影）。图 3.15 显示了使用阴影的效果。

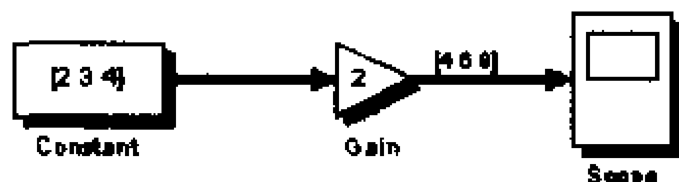


图 3.15 阴影的效果

3.4 库的概念及操作

因为有了库，用户就可以把模块从外部库中复制到自己的模型内，并且在源模块变化后，还可以使复制的模块自动更新。对那些开发自己的模块库或者使用其他模块库（如模块集）的用户来说，使用库可以确保在自己的模型中自动地包含这些模块的最新变化。

3.4.1 术语

首先，让我们了解一些非常重要的术语。

(1) 库：模块的集合。一个库必须是通过 File 菜单上的 New Library 命令或在 MATALB 命令窗口上输入的 new_system 命令所创建出的模块集合。

(2) 库模块：在库中的模块。

(3) 参考模块——库模块的复本。

(4) 链接：参考模块与它的库模块之间的连接，它给予 Simulink 在库模块变化时更新参考模块的渠道。

(5) 复制：从一个库模块或另一个参考模块到创建一个新的参考模块的操作。

图 3.16 说明了这些术语的含义和相互关系。

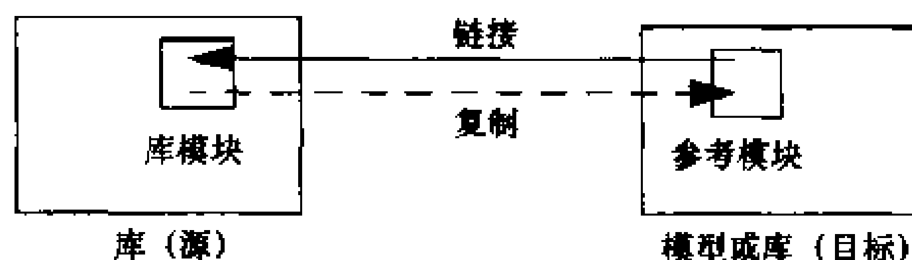


图 3.16 术语间的相互关系

3.4.2 创建库

在 File 菜单的 New 子菜单中选择 Library 项，就可以创建一个库。Simulink 将显示一个标题为 Library:untitled 的新窗口，如果已经存在一个无标题的窗口，则这个标题后会自动附加一个序号。也可以在命令行中使用如下命令创建一个库：

```
>>new_system( 'newlib', 'Library')
```

这个命令将创建一个名为'newlib'的新库。要显示该库，应使用 open_system 命令。这些命令的使用方法见第 2 章。

从一个库中拷贝模块之前，这个库必须被命名(保存)。

3.4.3 修改库

当打开一个库时，它会自动锁定，不允许用户修改它的内容。要解除库的锁定状态，可以在 Edit 菜单中选择 Unlock Library 项。

关闭库窗口时又会锁定该库。

3.4.4 向模型中复制库模块

通过复制和粘贴命令，或通过从库窗口向模型窗口拖动模块(参阅 3.3.3 一节)，或者通过从库浏览器向模型窗口拖动模块(参阅“浏览模块库”一节)，都可以从库中把一个模块复制到模型里。

当用户把一个库模块复制到一个模型或者另一库中时，Simulink 将创建一个指向该库模块的链接。参考模块就是该库模块的复本。用户可以在参考模块中修改模块参数，但不能封

装该模块；如果它已经是被封装过的，则不能编辑该封装。同时，不能设定参考模块的调回参数。如果要查看参考模块的封装，Simulink 就将显示该库模块的底层系统。

库与其参考模块是通过模块名链接的，这就是说，参考模块指向特定的模块和库，它们的名字在复制时就生效了。

Simulink 在试图更新参考模块时，如果在 MATLAB 路径上既找不到库模块也找不到源库，那么库及其参考模块之间的链接就是未判定的(unresolved)，Simulink 会提示出错信息并用红虚线显示这些模块。出错信息是：

```
Failed to find block "source-block-name"
in library "source-library-name"
referenced by block
"reference-block-path".
```

(未从模块“参考模块路径”引用的库“源库名”中找到模块“源模块名”)



图 3.17　未判定模块示意

未判定参考模块如图 3.17 所示(红色)。

要修复一个破坏了的链接，可以选择下面几种方法之一：

(1) 删除未正确链接的参考模块，重新将库模块复制到模型中。

(2) 给 MATLAB 路径增加包含所需源库的目录，然后在 Edit 菜单中选择 Update Diagram 项。

(3) 双击参考模块，在出现的对话框里修改路径名并单击 Apply 或 Close。

所有模块都有一个说明该模块是否为参考模块的链接状态(LinkStatus)参数。该参数可以有下面这些值：

(1) 'none'表示该模块不是一个参考模块。

(2) 'resolved'表示该模块是一个参考模块并且链接已判定有效。

(3) 'unresolved'则表示该模块是一个参考模块但未判定有效的链接(未链接)。

3.4.5　更新已链接的模块

Simulink 在下列场合更新模型中或者库中的过时参考模块：

(1) 在载入模型或者库时。

(2) 在 Edit 菜单中选择 Update Diagram 项或者运行该仿真时。

(3) 在使用 get_param 命令查询模块的 LinkStatus 参数时(参阅“获得库模块信息”一节)。

(4) 在使用 find_system 命令时。

3.4.6　断开与库模块的链接

可以断开一个参考模块与它的库模块之间的链接，使该参考模块变成源库模块的一个简单的复本，对库模块的更改将不再影响该模块。断开与库模块的链接，可以将模型独立地应用，而不需要借助库。

要断开参考模块与它的库模块之间的链接，应选中模块，然后在 Edit 菜单中选择 Break Library Link 项。同样也可以使用命令，如利用如下命令将模块的 LinkStatus 参数改变为

'none'：

```
>>set_param('refblock','LinkStatus','none')
```

还可以使用如下命令保存一个系统并断开参考模块与库模块之间的所有链接：

```
>>save_system('sys','newname','BreakLinks')
```

3.4.7 为参考模块寻找库模块

选中参考模块，然后在Edit菜单中选择Go To Library Link项，就可以为参考模块寻找到与其链接的源库和源模块。如果库是打开的，Simulink就选中该库模块(在模块上显示选中句柄)并将源库设定为当前窗口；如果库是关闭的，则Simulink先将其打开，然后再选中库模块。

3.4.8 获得库模块信息

可以使用libinfo命令来获得参考模块在系统中的有关信息。命令格式为：

```
>>libdata = libinfo(sys)
```

其中sys是系统的名字。该命令返回 $n\times1$ 结构矢量，n 为sys中库模块的数目。每个结构矢量都有四个域：

(1) Block，模块路径。

(2) Library，库名。

(3) ReferenceBlock，参考模块路径。

(4) LinkStatus，链接状态，为"已判定(resolved)"或者"未判定(unresolved)"。

3.4.9 浏览模块库

库浏览器可以使用户快速定位并向模型中复制库模块。

编者提示：库浏览器仅在Windows平台上可用。

在库浏览器中定位模块，既可以利用库浏览器树，也可以使用库浏览器的搜索功能。

1. 库浏览器树定位

库浏览器树显示安装在系统中所有的模块库的列表。用户可以查看或者隐藏库的内容，通过使用鼠标或键盘来展开或折叠树。要展开/折叠树，单击库名项旁边的[+]/[−]按钮，或者选择一库名项，按键盘上的[+]/[−]键或右/左箭头键。上/下箭头键可以上下移动。

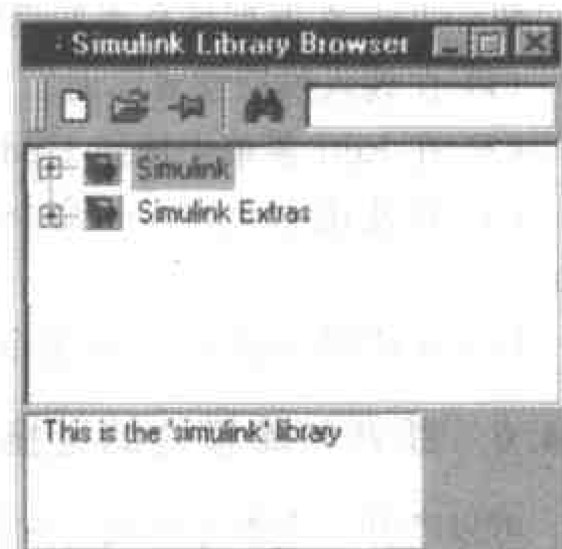


图3.18 库浏览器

2. 搜索库

寻找一个特别的模块，在库浏览器Find按钮旁边的编辑框内输入该模块的名字，然后单击Find按钮。

3. 打开一个库

在浏览器里的用鼠标右键单击欲打开库名，此时显示一个Open Library按钮，单击它就可打开该库。

4. 创建和打开模型

创建一个模型时，单击库浏览器工具栏上的 New 按钮。欲打开一个已存在的模型，单击工具栏上的 Open 按钮。

5. 复制模块

要从库浏览器中复制一个模块到一个模型，可选中浏览器中的模块，拖拉到模型窗口，放在需要的地方。

6. 显示一个模块的帮助信息

显示一个模块的帮助信息，可右键单击该模块，并选择随后弹出的按钮。

7. 固定库浏览器

为了保持库浏览器总是在其他窗口的上方，可选择浏览器工具栏上的 Push Pin 按钮。

3.5 线

线传递信号。每条线都能携带一个标量或矢量信号。线将一个模块的输出端口与另一模块的输入端口连接起来。使用分支线，还可以将一个模块的输出端口同多个模块的输入端口连接。

3.5.1 在两个模块间画线

连接一个模块的输出端口与另一模块的输入端口：

(1) 将光标置于第一个模块的输出端口边(不一定非要很精确地放到该端口)，此时光标形状变为十字状，如图 3.19 所示。

(2) 按下鼠标键不放。

(3) 拖拉鼠标指针到第二模块的输入端口处。可以把光标放置到端口或接近端口，或者置于模块上。如果把光标置于模块上，则线将与距离最近的输入端口连接。此时，光标变为双十字线，如图 3.20 所示。

(4) 放开鼠标键。此时，模块的端口标志消失，模块被一条线和一个代表信号流动的方向箭头连接在一起。连接线的生成可以从输出到输入，也可以从输入到输出。如图 3.21 所示的箭头是从输入端口开始画的，结果一样，信号也是相同的。

Simulink 画连接线使用水平和垂直线段，要画对角线，在画线时按下 shift 键即可。

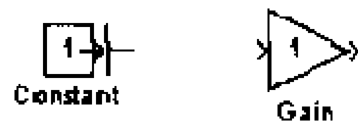


图 3.19 模块的连接示意 1

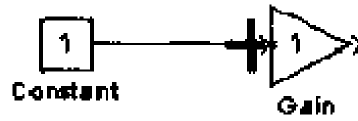


图 3.20 模块的连接示意 2

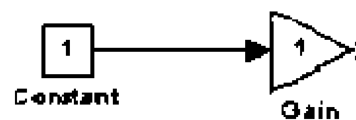


图 3.21 模块的连接示意 3

3.5.2 画分支线

分支线是从一现有的线上再画一条连接到某一个模块的输入端口线，现有的线和分支线携带的是相同的信号。使用分支线可以使一个信号送到多个模块。

在图 3.22 的例子里，Product 模块的输出同时加到 Scope 模块和 To Workspace 模块。

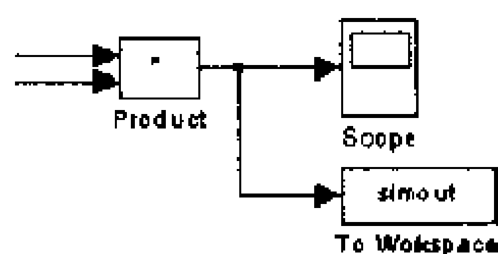


图 3.22 分支线示意

可以按下列步骤画分支线：

(1) 把鼠标指针放到想要开始画分支线的地方。

(2) 按下 Ctrl 键，按下鼠标左键不放。

(3) 将指针拖拉到目标模块的输入端口处，然后放开鼠标和 Ctrl 键。

还可以不用按下鼠标左键和 Ctrl 键，直接按下鼠标右键画分支线。

3.5.3 画线段

用户或许想画一条可以任意分段的直线，而不是 Simulink 所画的直线；或者，要复制已经画好连接线的模块，这些都可以通过画线段实现。

画一条线段，线的另一端悬空，并出现一箭头。想要续接另一条线段，将光标放到该线段悬空的一端，画另一条线段。Simulink 画出的线都是水平或垂直的线，要画对角线段，可在画线同时按下 Shift 键。

1. 移动线段

移动线段，可以执行下列步骤(图 3.23～3.26)：

(1) 把鼠标指针放在要移动位置线段上；

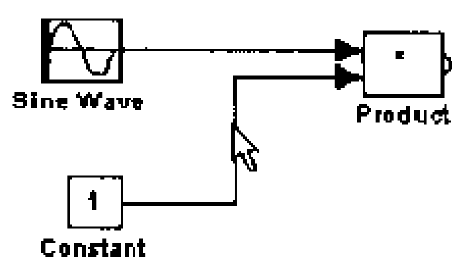


图 3.23 线的移动 1

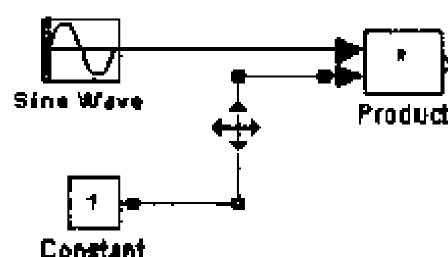


图 3.24 线的移动 2

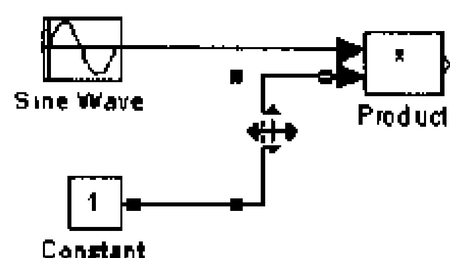


图 3.25 线的移动 3

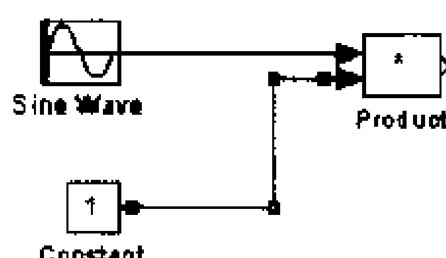


图 3.26 线的移动 4

(2) 按下鼠标左键不放；

(3) 拖拉线段到想要的位置；

(4) 放开鼠标键。

不能移动直接连接在模块端口上的线段。

2. 把一条线分成段

把一条线分成两个段，保持线的端点在它们的原始位置，且加入一个顶点。把一条线分段，执行下列步骤(图 3.27～3.31)：

(1) 选中该线；

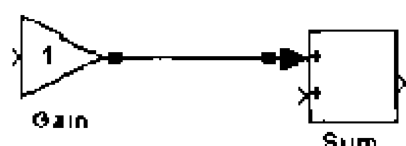


图 3.27　将一条线分段示意 1

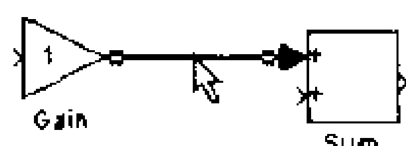


图 3.28　将一条线分段示意 2

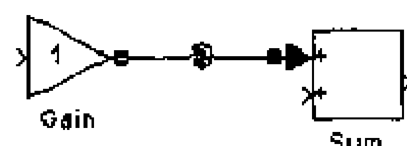


图 3.29　将一条线分段示意 3

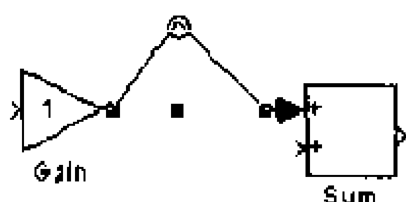


图 3.30　将一条线分段示意 4

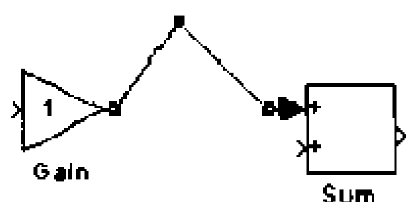


图 3.31　将一条线分段示意 5

(2) 把鼠标指针放在要加入顶点的位置；

(3) 在按下 Shift 键的同时，按下鼠标左键不放。光标形状变为一个围绕新的顶点的圆周；

(4) 拖拉指针到想要到的位置；

(5) 放开鼠标键和 Shift 键。

3. 移动线的顶点

移动线的顶点，执行下列步骤(图 3.32～图 3.34)：

(1) 把指针放在顶点处，然后按下鼠标左键不放。光标变为一个围绕顶点的圆周；

(2) 拖拉指针到想要到的位置；

(3) 放开鼠标键。

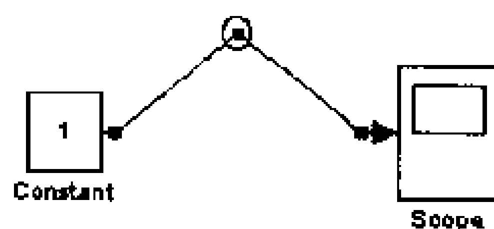


图 3.32　移动线的顶点示意 1

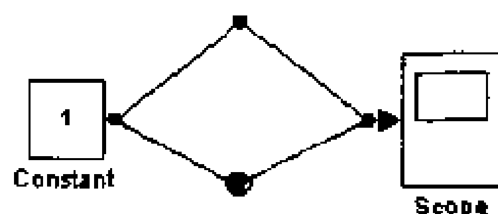


图 3.33　移动线的顶点示意 2

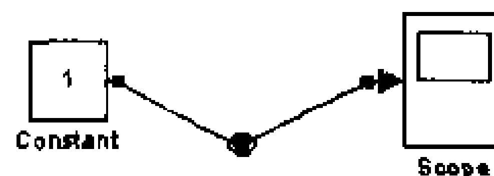


图 3.34　移动线的顶点示意 3

3.5.4　显示线的宽度

从 Format 菜单中选择 Vector Line Widths，就能够显示一个模块中矢量的宽度线，Simulink 在接收信号的模块和发出信号的模块处显示各矢量信号的宽度(维数)。通过从 Format 菜单中选择 Vector Line Widths，就可使矢量线以宽线显示。

当开始仿真或者更新图表时，若 Simulink 检测出输入端口与输出端口不匹配，则显示一

出错信息，并显示线宽。

3.5.5　在线上插入模块

可以在一条线上插入一个模块，插入的位置是放置模块的地方。

编者提示：需要插入的模块必须只有一个输入端口和一个输出端口。

插入模块步骤如下(图3.35～图3.37)：

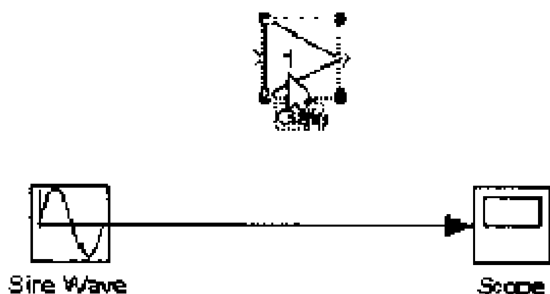


图3.35　线上插入模块步骤1

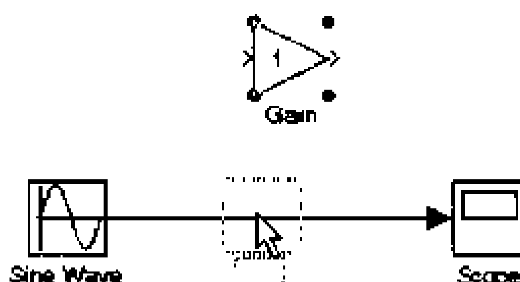


图3.36　线上插入模块步骤2

(1) 把光标放在要插入的模块上，并按下鼠标左键；

(2) 拖动模块到想要插入的线上；

(3) 放开鼠标键，模块就插入在放开鼠标的线上。

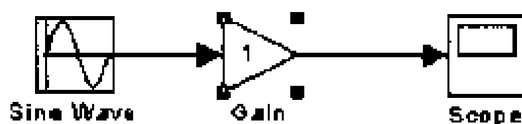


图3.37　线上插入模块步骤3

3.5.6　信号标签及其编辑和传递特性

在复杂的模型中，经常需要使用信号标签来标注线上信号。对水平线或水平线段来说，标签可以出现在上方或下方；对于垂直线或垂直线段，则在左边或右边。标签可以选择靠近任一端，或者居中，多个标签的位置可以选择在不同方位显示。

1. 插入信号标签

创建信号标签，只须双击线段，在光标出现的插入点处输入标签内容。当在其他地方点击鼠标，标签的位置就会固定。

编者提示：在创建信号标签时，要在线上双击，否则创建的是模型注释。

2. 移动信号标签

把标签拖到新位置即可，放开鼠标键时，标签的位置会在该线附近固定。

3. 复制信号标签

需拷贝一个信号标签时，在拖动标签的时候按下Ctrl键。放开鼠标键时，新位置就会出现一个标签，并不影响原标签。

4. 修改信号标签

修改信号标签可能有不同的目的：

(1) 替换标签：在该标签上单击，然后双击或者拖动光标选中整个标签，然后，输入新标签。

(2) 在标签中插入字符：在插入位置单击，然后插入字符。

(3) 替换字符：拖动鼠标选中要替换的字符，然后输入新字符。

5. 删除标签

(1) 删除信号标签:选中标签,删除所有标签中的字符,然后单击标签外部。

(2) 删除部分标签字符:在选择部分标签字符时按下 shift 键,然后按下 Delete 或者 Backspace 键。

6. 改变信号标签的字体

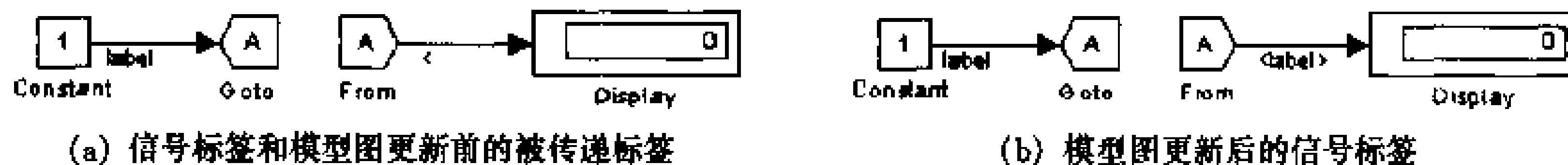
选中带有信号标签的线,从 Format 菜单中选择 Font 项,然后从 Set Font(设定字体)对话框选择一种字体。

7. 信号标签的传递

信号标签传递是指给从模块发出的连接线自动添加标签,也就是说模块将输入线的已有信号标签自动传递给它的输出线。支持信号标签传递的模块有 Demux、enable、Inport、Mux、Selector 和 Subsystem 模块。

为传递一个信号标签,须给连接模块的输出线创建一个以"<"符号开始的标签。当运行该仿真或更新模型时,实际的信号标签就出现在线上,并括在尖括号中。这个实际的信号标签就是连接模块遇到的第一个信号标签。

图 3.38 的例子说明了一个模型使用信号标签和传递标签的情形,两幅图分别是更新前和更新后的图示。在图(a)中,输入 Goto 模块的信号标签为 Lable, From 模块输出信号的关联标签为<。图(b)显示了相同模型在从 Edit 菜单中选择 Update Diagram(更新模型)项后标签的变化。



(a) 信号标签和模型图更新前的被传递标签　　(b) 模型图更新后的信号标签

图 3.38　信号标签更新前后

图 3.39 的例子显示了信号为向量时标签传递的情况。该图是更新模型图后的显示。

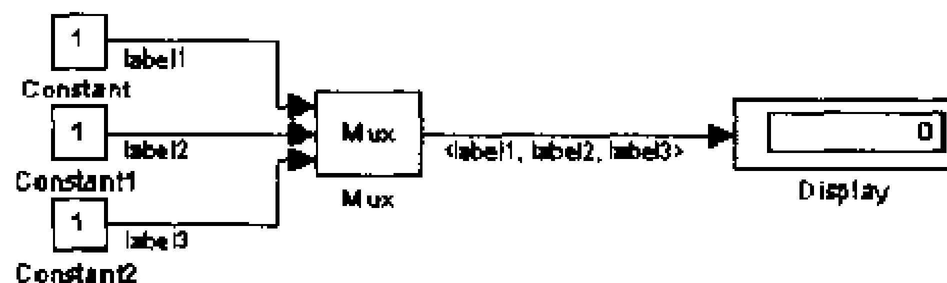


图 3.39　矢量输入时的标签传递

3.5.7　信号属性及其设置

可以通过 Simulink 信号属性对话框来查看或设置信号的属性。选中携带该信号的线,从 Edit 菜单中选 Signal Properties 就可显示对话框。

如图 3.40 所示的对话框包含以下属性。

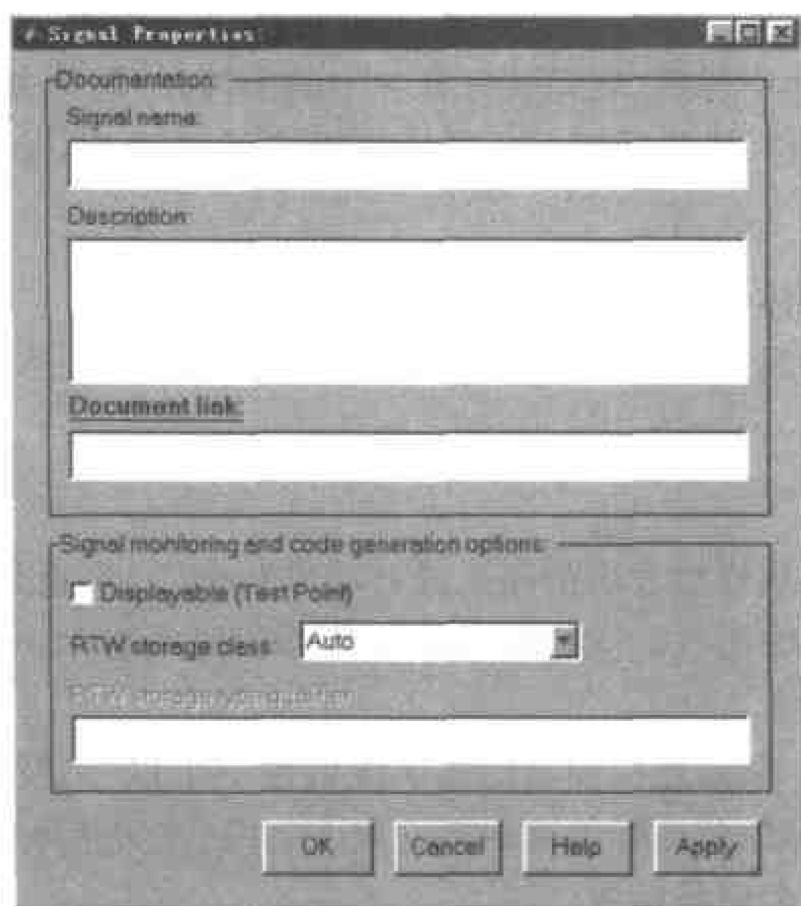


图3.40 信号属性对话框

(1) Signal Name:信号名;

(2) Description:输入信号描述;

(3) Document Link:输入一个MATLAB的表达式以指向该信号的记录文档,单击字段标签(即,Document Link),显示该记录文档。例如,输入表达式:

```
>>web(['file:///' which('foo_signal.html')])
```

当单击字段标签时,将用MATLAB默认的Web浏览器来显示foo_signal.html文件;

(4) Displayable (Test Point):测点显示复选框。选中则表示在仿真过程中,该信号能被显示。

编者提示:下面两个属性通常在Real-Time Workshop生成的模型代码时才需要设置,而且只有当系统装载Real-Time Workshop才会显示。如果不打算生成模型代码,可以忽略它们。

(5) RTW Storage Class。详情请参看"Real-Time Workshop用户指南"或列表内容解释。

(6) RTW Storage Type qualifier。参看"实时工作间用户指南"以获取更多的信息。

3.6 注释

注释提供了关于模型的文本信息。用户可以在模型图表中任意空白处添加注释(图3.41)。

3.6.1 创建模型注释

在模型图中的空白处双击,会出现一个小的矩形框,且光标变为在插入点闪烁,可键入注

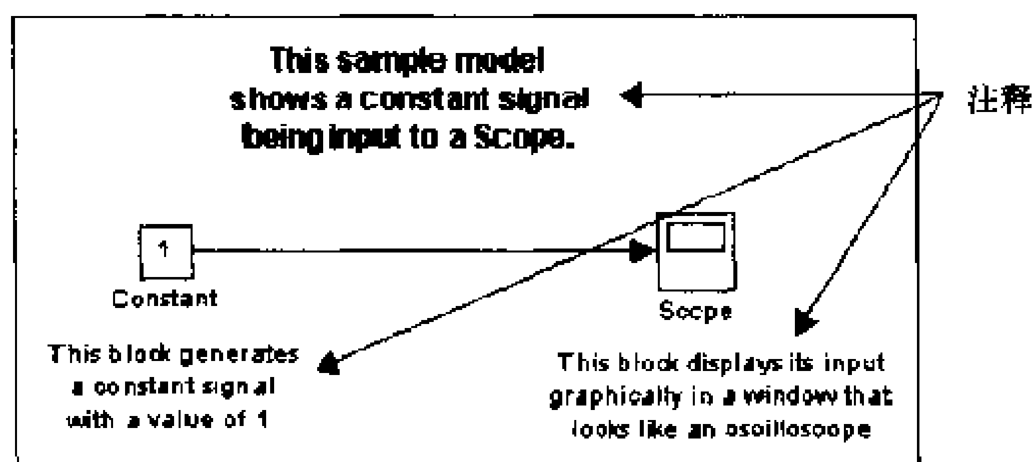


图 3.41 注释的位置

释内容。每一行都是居中设置。

3.6.2 移动注释

拖拉它到新位置即可。

3.6.3 编辑注释

(1) 更新注释:在 WINDOWS 或 UNIX 系统下,单击该注释,然后双击或者拖动光标选中它,接着输入新注释。

(2) 插入字符:在想要插入的位置单击,输入字符。

(3) 替换字符:拖拉鼠标选中要替换的字符范围,然后输入新文本。

3.6.4 删除注释

先按下 shift 键,并选中该注释,然后按 Delete 或 Backspace 键。

3.6.5 更改注释的字体

选中要更改的注释内容,然后从 Format 菜单选择 Font 项,从对话框中选择一种字型和尺寸。

3.7 数据类型

数据类型(Data Type)指的是计算机在存储器中表示数的方式。确定了数据类型,就决定了分配给该数的存储量,将数据值编码为二进制数格式的方法以及用于该类型数据的操作等。大多数的计算机提供数据类型的选择,每个类型都有具体的各自优点,如在精度、动态范围、性能及存储量等方面。为便于利用各个数据类型的优点来优化 MATLAB 程序性能,MATLAB 允许用户指定变量的数据类型。同样 Simulink 也允许用户指定 Simulink 信号和模块参数的数据类型。

指定模型的信号和模块参数的数据类型,对实时控制应用方面特别有用。例如,它允许在由自动代码产生工具生成的代码中,指定 Simulink 模型的信号和模块参数的最佳数据类型。例如 MathWorks 中的 Workshop,通过给模型的信号和参数选择最适当的数据类型,可以大大提高模型性能并减小模型的代码冗余。

Simulink在一个仿真运行之前和运行过程中，执行全局检验以确保模型"typesafe"(类型安全)，也就是说，模型的代码将不会产生溢出而导致不正确的结果。使用Simulink默认数据类型(双精度型)的模型可以说是绝对安全的。因此，如果用户并不打算生成模型代码或在自己的模型中使用一个非默认的数据类型，可以跳过本节余下部分的阅读。

相反，如果要生成模型代码或在模型中使用非默认的数据类型时，需要认真地阅读本节余下的部分，特别是关于"数据类型规则"的内容。这样，可以避免由于数据类型错误而导致模型不能运行或不能完成仿真。

3.7.1 Simulink支持的数据类型

Simulink支持所有MATLAB的内置数据类型。"内置数据类型"是指与由MATLAB用户自己定义的数据类型相对的由MATLAB本身定义的数据类型。除非特别指出，Simulink文献中提到的数据类型都指的是内置数据类型。下面的表3.2列出了MATLAB内置数据类型。

表3.2 系统支持的数据类型

名	描 述
double	双精度浮点
single	单精度浮点
int8	单精度8位整型
uint8	无符号8位整型
int16	单精度16位整型
uint16	无符号16位整型
int32	单精度32位整型
uint32	无符号32位整型

除了内置类型之外，Simulink还定义了一种布尔值(1或0)类型，它在内部实际是由uint8值表示。

3.7.2 支持数据和数字信号类型的模块

默认时，所有Simulink模块接受的信号数据类型为双精度型。一些模块也接受布尔值输入，一些模块支持多数据类型的输入。表3.3列出了支持布尔值和多数据类型的Simulink模块，同时也列出了支持复杂信号的模块。

更多的信息参见第7章中有关特定模块的参数、输入和输出数据类型。如果文献中没有指定模块数据类型，则该模块输入或输出的数据类型均为双精度型。

表 3.3 支持布尔值的多数据类型的模块

模块	简 介
Abs	输入双精度型的实数或复数信号，输出双精度型的实信号
Combinatorial Logic	如果布尔方式被激活(参见“启用严格布尔值检查”)，输入和输出数据类型都是布尔值；否则为双精度型
Constant	输出任意类型的实数或复数信号
Data Type Convertion	输入和输出任意实数或复数数据类型
Demux	接受混合类型信号矢量
Display	接受任意类型的实数或复数信号
Dot Product	输入和输出双精度型的实数或复数的数据
Enable	相应的子系统使能端口接受布尔值或者双精度型
From	输出连接到相应的 Goto 模块的数据类型
From Workspace	输出相应的工作空间的类型值
Gain	输入可以是除布尔型以外的任意类型的实数或复数信号或者矢量
Goto	输入可以是任意类型
Ground	输出为与输入端口相连的信号类型相同的 0 值信号
Hit Crossing	输入双精度型信号。如果布尔值方式被激活，则输出布尔值(参看“激活严格布尔值检查”)；否则，为双精度型
Inport	接受任意型的实数或复数信号。如果该 inport 是根端口，或直接连接到一个同一个子系统的输出口，输入矢量信号的各元素必须是同一类型
Integrator	积分器模块接受和输出双精度型的信号，它的外部置位端口接受双精度或布尔型数据
Logical Operator	如果布尔值方式被激活(参看“激活严格布尔值检查”)，输入和输出布尔类型的实数信号；否则，为双精度型的信号
Manual Switch	接受任意类型的实数或者复数信号。所有的输入必须有相同的信号和数据类型
Math Function	输入和输出双精度型实数或复数数值
MATLAB Function	输入和输出双精度型实数或复数数值
Memory	输入任意数据类型的实数或复数信号
Merge	输入和输出任意实数或复数的数据类型
Multiport Switch	该模块的控制输入端口接受一个除布尔型外的任意类型的实数信号。其他输入端口接受任意类型的实数或复数信号。所有的输入信号必须具有相同的数字或数据类型。模块的输出信号的类型由输入信号类型决定

续表 3.3

模块	简　介
Mux	每个端口接受所有 Simulink 支持的数据类型，包括混合型矢量
Out	接受 Simulink 的任意数据类型作为输入。如果输出口是个子系统并且无指定初始条件时，输入可以是混合型矢量
Product	接受除布尔值外的任意类型的实数或复数值信号。所有输入必须具有相同的数据类型
Relational Operator	接受任意 Simulink 支持的数据类型作为输入，两个输入必须具有相同的类型。如果布尔值方式被激活（参看“激活严格布尔值检查”），输出为布尔值；否则输出为双精度型
Rounding Function	接受和输出双精度型实数或复数数据
Scope	接受任意类型的实数或复数数据
Selector	输出选中输入信号的数据类型
Sum	接受任意 Simulink 的数据类型作为输入。所有的输入必须具有相同的类型。输出与输入的数据类型相同
Switch	接受任意类型实数或复数信号作为转换器的输入（输入 1 和 3）。两个转换输入必须具有相同的类型。模块的输出信号与输入数据类型相同。阈值输入的类型必须是布尔型或双精度型
Terminator	接受任意 Simulink 的类型
To Workspace	接受任意 Simulink 的数据类型作为输入
Trigger	相应的子系统控制口接受布尔型或双精度型的数据
Trigonometric Function	输入和输出任意双精度型的实数或复数信号
Unit Delay	接受和输出任意类型的实数或复数信号
Width	接受任意类型的实数或复数信号，包括混合矢量信号
Zero-Order Hold	接受任意 Simulink 数据类型作为输入

3.7.3 指定模块参数的数据类型

当输入模块参数的数据类型是可由用户指定的时，可以使用下面命令指定参数：

```
type(value)
```

其中，type 是数据类型名，value 是参数值。表 3.4 举例说明了命令用法。

表 3.4 type(value)的用法

精 度	含 义
single(1.0)	将 1.0 指定为一个单精度值
int8(2)	将 2 指定为一个 8 位整数型值
int32(3+2i)	将一个复数指定为一个实部和虚部是 32 位整数型

3.7.4 创建一个指定数据类型的信号

用户可以把任意指定数据类型的信号引入到模型中，请选择如下任意一种途径：

(1) 通过最前级输入端口或者一个 From Workspace 模块，将想要的数据类型的信号从 MATLAB 工作空间加载到用户的模型中。

(2) 在用户模型中创建一个 Constant 模块，将它的参数指定成想要的数据类型。

(3) 使用数据类型转换(Data Type Conversion)模块将信号转换成想要的数据类型。

3.7.5 显示端口数据类型

从 Simulink 的 Format 菜单选择 Port Data type，可在模型中显示端口的数据类型。当改变模型图上各个元素的数据类型后，Simulink 并不更新端口数据类型显示。要更新显示，键入 Ctrl—D。

3.7.6 数据类型传递

当运行一个仿真，激活端口数据类型的显示，或更新端口显示时，Simulink 执行一个叫做数据类型传递的处理步骤。这个步骤包括确定信号类型没有在别处被指定，确定检验信号与输入端口的类型以保证不会发生冲突。如果出现类型冲突，则会出现一个出错对话框，指出数据类型出现冲突的信号与端口，并将引起冲突的信号路径设置成高亮显示状态。

编者提示：可以将 typecasting(数据类型转换)模块插入自己的模型中以解决类型冲突。详见“转换信号数据类型”。

3.7.7 数据类型规则

熟悉下面的规则将有助于构建“类型安全”的模型，以免执行出错：

(1) 信号数据类型通常不影响参数的数据类型，反之亦然。

一个不能忽视的例外是：Constant 模块的输出数据类型由其参数的数据类型决定。

(2) 如果模块的输出是某输入和参数的函数，且输入类型和参数类型不同，则在计算输出之前，Simulink 将该参数的类型转换为与输入类型一致。详见“转换信号的数据类型”。

(3) 一般情况下，模块输出数据类型与输入的数据类型是一致的。但 Constant 模块和数据类型转换模块的输出数据类型由模块参数决定。

(4) 虚拟模块接受任意类型的信号作为输入。虚拟模块的例子包含 Mux 和 Demux 模块及无条件执行子系统。

(5) 连接到非虚拟模块端口的矢量信号的各元素必须具有相同的数据类型。

(6) 连接到非虚拟模块输入端口的信号不能是不同的数据类型。

(7) 控制端口(例如:Enable port 和 Trigger port)接受布尔型或双精度型的信号。

(8) Solver 模块只能接受双精度型信号。

(9) 若将一个不是双精度型的信号作为一个模块的输入,则该模块不能进行过零检测。

3.7.8 激活严格布尔类型检测

默认情况下,当将双精度型的信号连接到需要布尔型输入的模块时,Simulink 只是检测但并不提示错误。这是考虑到与 Simulink 早期版本中只支持双精度型数据模块的兼容性。用户可以启用严格的布尔类型检测,在 Simulink Parameters 对话框的 Diagnostics 页取消选中 Relax Boolean type checking(参见第 5 章"诊断页")。

3.7.9 转换信号的数据类型

当一个信号连接到一个不能接受该信号类型的模块上时,Simulink 将提示出错信息。如果用户想要创建这样一种连接,必须明确将信号类型转换为模块能接受的类型。可以使用 Simulink Data Type Conversion(数据类型转换)模块来实现这样的转换(参见第 7 章"Data Type Conversion 模块)。

3.7.10 转换参数的数据类型

在通常情况下,在仿真运行期间,当模块输出是某输入信号和参数的函数时,如果模块参数与信号的数据类型不同,则 Simulink 会自动将模块参数的数据类型转换为信号的数据类型。但此规则有下列例外:

(1) 如果信号的数据类型不同于参数,Simulink 会停止仿真,并显示出错信息。例如图 3.42 所示的模型。

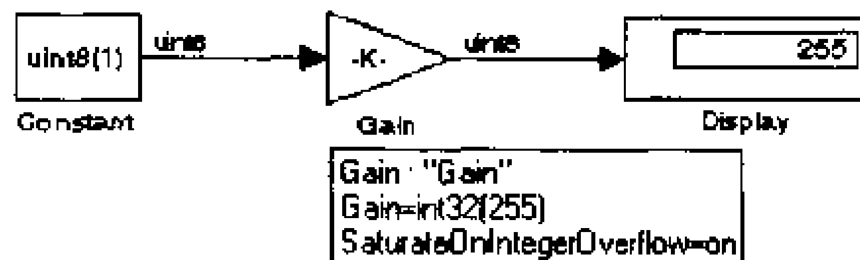


图 3.42 模型组成

这个模型使用一个 Gain 模块来放大一个常数输入信号。计算 Gain 模块的输出结果,需要计算输入信号与增益的积,这样的计算要求两个值的数据类型必须相同。然而,在本例中,信号的数据类型是 uint8(无符号 8 位字),而增益参数的数据类型是 int32(有符号 32 位整型)。因此计算增益模块输出必须进行类型转换。

当进行这样的转换时,Simulink 总是改变参数类型与信号类型一致。这样,在这个例子中,必须将 Gain 模块的增益值的数据类型转换成与输入信号类型相一致,而只有当输入信号的数据类型(uint8)能够表示该增益值时才能进行转换。对于本例,Simulink 能进行这种转换,是因为增益值 255,在 uint8 数据类型的范围(0 到 255)内。因此,模型的仿真不出现错误。然而,如果增益值稍微大一点(如 256),在运行仿真后,就会提示溢出范围的错误。

(2) 如果信号数据类型除了牺牲精度外还能表示参数,Simulink 会发布一则警告信息并

继续仿真。参考图3.43所示的模型。

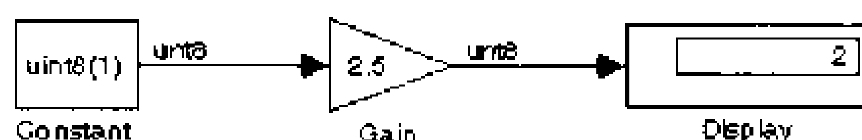


图3.43 另一个模型

在这个例子中，信号的类型只能适应整数值，而增益值有分数部分。运行仿真时，Simulink将增益值取整到最近的整数值(2)，并提示精度损失的警告。另一方面，如果增益值是2.0，仿真时不出现警告信息，因为在转换时没有引起精度损失。

编者提示：将一个int32参数转换为浮点或双精度型，均会引起精度损失。如果参数值很大，损失可能是很严重的。如果int32型参数的转换确实引起了精度损失，Simulink会提出警告信息。

3.8 复信号工作方式

默认情况下，信号值都是实数，然而，模型是能够创建和操作复数信号值的。

通过下列任意一种途径，均可以把一个复数值信号引入模型。

(1) 通过最前级输入端口把复值信号从MATLAB工作空间加载到模型中。

(2) 创建一个Constant模块并把它的值设置为复数值。

(3) 创建相应于复信号实部和虚部的两个实信号，然后利用Real-Image to Complex转换模块将两部分合并成一个复信号。

用户可以通过模块操作复信号。大多数的Simulink模块能够接受复信号作为输入。如果不能确定一个模块是否能接受复信号，请参看第7章。

3.9 鼠标和键盘操作简介

下面这些表格总结了使用鼠标和键盘来操作模块、线及信号标签的要点。LMB表示按鼠标左键，CMB表示按鼠标中键，RMB表示按鼠标右键。

1. 用鼠标和键盘对模块的操作

表3.5 操作模块

任务	Windows系统	UNIX系统
选中模块	LMB	LMB
选中多个模块	Shift－LMB	Shift＋LMB；或只按CMB
从其他窗口复制模块	拖拉模块	拖拉模块
移动模块	拖拉模块	拖拉模块
复制模块	Ctrl－LMB并拖动；或RMB并拖动	Ctrl＋LMB并拖动；或RMB并拖动

续表 3.5

任务	Windows 系统	UNIX 系统
连接模块	LMB	LMB
断开模块连接	Shift+拖动模块	Shift+拖动模块；或CMB并拖动

2. 用鼠标和键盘对线的操作

表 3.6 操作线

任务	Windows 系统	UNIX 系统
选中线	LMB	LMB
选中多条线	Shift+LMB	Shift+LMB；或只按CMB
画分支线	Ctrl+拖动线；或RMB并拖动线	Ctrl+拖动线；或RMB+拖动线
绕过模块画线	Shift+画线段	Shift+画线段；或者CMB并画线段
移动线段	拖动线段	拖动线段
移动箭头	拖动箭头	拖动箭头
创建线段	Shift+拖动线	Shift+拖动线；或CMB+拖动线

3. 用鼠标和键盘对信号标签的操作

表 3.7 操作信号标签

任务	Windows 系统	UNIX 系统
创建信号标签	双击线，然后输入标签	双击线，然后输入标签
复制信号标签	Ctrl+拖动标签	Ctrl+拖动标签
移动信号标签	拖动标签	拖动标签
编辑信号标签	单击标签，然后编辑	单击标签，然后编辑
删除信号标签	Shift+单击标签，然后按Delete	Shift+单击标签，然后按Delete

4. 用鼠标和键盘对注释的操作

表 3.8 操作注释

任务	Windows 系统	UNIX 系统
创建注释	双击图表，然后输入文本	双击图表，然后输入文本
复制注释	Ctrl+拖动注释标	Ctrl+拖动注释标
移动注释	拖动注释标	拖动注释标
编辑注释	单击文本，然后编辑	单击文本，然后编辑
删除注释	Shift+单击注释，然后按Delete	Shift+单击注释，然后按Delete

3.10 创建子系统

随着模型变大和复杂性的增加，可以把几个模块组合成子系统。使用子系统有下面这些优点：

(1) 减少模型窗口上显示的模型数目。

(2) 将功能相关的模块组合在一起。

(3) 建立一种层次型的模型图表，子系统模块在一层，组成子系统的模块在另一层。

下面介绍两个创建子系统途径。

3.10.1 通过添加 Subsystem 模块创建子系统

先创建子系统，在模型中加入 Subsystem 模块，然后再添加子系统的各个组成模块，形成子系统：

(1) 从 Sibnal&Systems 库中把 Subsystem 模块拷贝到模型中。

(2) 双击打开该 Subsystem 模块。

(3) 在空白的 Subsystem 窗口内添加子系统的内容模块，用 Inport 模块来代表该子系统外部的输入，用 Outport 模块来代表子系统的输出。例如，图 3.44 所示的子系统包含一个 Sum 模块，及 Inport 和 Outport 模块，它们分别代表子系统的输入和输出。

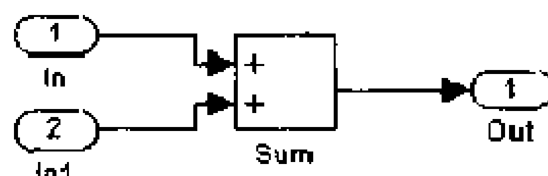


图 3.44 子系统组成

3.10.2 通过组合已有的模块创建子系统

如果用户模型中已经包含要组合成子系统的模块，只须组合这些模块建立子系统。

(1) 用方框把要组合的模块和连接线包围起来。注意不能分别选中各个模块或使用 Select All 命令。更多信息参见本章“选择对象”一节。

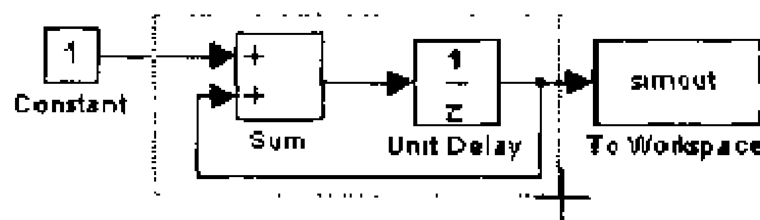


图 3.45 组合创建子系统

例如，图 3.45 显示了一个表示计数器的模型，已用方框把 Sum 模块和 Unit Delay 模块框在一起。当放开鼠标键时，两个模块和所有的连接线被选中。

(2) 从 Edit 菜单中选择 Creat Subsystem 项，选中的模块被一个 Subsystem 模型图标代替。图 3.46 显示选择了 Creat Subsystem 命令后的情况(并调整该 Subsystem 模块大小，便于端口标签显示)。

双击打开 Subsystem 模块，将显示该子系统的内层结构，如图 3.47 所示。注意 Simulink 自动添加了 Inport 和 Outport 模块，代表该子系统从外部模块的输入和输出。

和其他所有模块一样，可以改变子系统模块的名字，也可以使用封装特性为子系统定制图标和对话框，详见第 6 章。

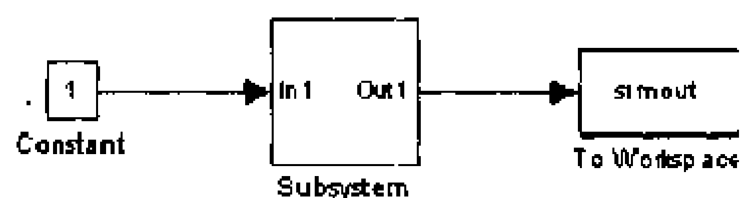


图 3.46　执行 Creat Subsystem 命令后的情况

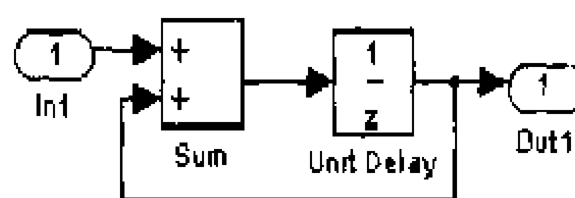


图 3.47　Subsystem 的内部结构

3.10.3　子系统端口标签

Simulink 在 Subsystem 模块端口标注标签，这些标签分别是 Inport 和 Outport 的模块名，子系统就是通过这些端口与外部模块连接的。

要隐藏端口标签，先选中该子系统模块，然后从 Format 菜单中选择 Hide Port Lables 项。还可以隐藏一个或几个端口的标签，选中要隐藏的特定的 Inport 或 Outport 模块，然后从 Format 菜单中选择 Hide name 项。

图 3.48 显示的是两个模型图标，左边的子系统包含两个 Inport 模块和一个 Outport 模块，右边的图显示了带有端口标签的子系统模型图标。

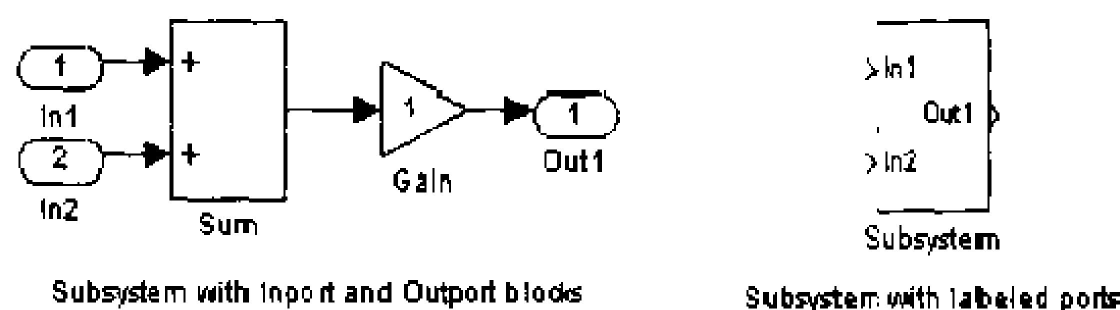


图 3.48　带有端口标签的子系统

3.10.4　使用调回例程

用户可以自己定义 MATLAB 表达式(程序)，此表达式在对模型或者一个模块以某种特殊途径进行操作时执行。这些表达式叫做调回例程，它与模块或模型的参数有关。例如，当一个调回与模块的 OpenFcn 参数关联时，用户双击模块名或者路径变化，调回例程就会执行。

定义调回例程并将它们与参数关联，使用 set_param 命令(参看第 2 章"set_param 命令")。

例如，当用户在双击 mymodel 模型中的 Test 模块时，下面的命令就会对变量 testvar 的值进行计算(模型必须为打开状态，并且变量已经定义)。

```
>>set_param('mymodel/Test','OpenFcn','testvar')
```

读者如果感兴趣，可以考察 clutch 系统(clutch.mdl)，会发现其中有许多与模型调回有关的程序。

下表列出用于用户定义调回例程的参数及这些调回程序的执行时机。调回操作一经发生例程就立刻执行。

表 3.9 模型调回参数

模型调回参数	执行时机
CloseFcn	模型框图关闭前执行
PostLoadFcn	载入模型后执行。给这个参数定义调回例程对产生一个界面非常有用，而该界面需要模型已经载入
InitFcn	模型仿真开始时执行
PostSaveFcn	模型被保存后执行
PreLoadFcn	模型载入前执行。给这个参数定义调回例程对载入模型变量非常有用
PreSaveFcn	模型被保存前执行
StartFcn	仿真开始前执行
StopFcn	仿真停止后执行。在 StopFcn 执行前，输出写到工作空间变量和文件

表 3.10 模块调回参数

模块调回参数	执行时机
CloseFcn	当用 close_system 命令关闭模块时执行
CopyFcn	在模块被复制好后执行。这个收回参数对于 Subsystem 模块是递归的，也就是说，如果一个子系统模块中包含已经定义了 CopyFcn 参数的模块，当子系统被复制时，该例程也被执行。另外，如果用 add_block 命令复制模块，该例程也执行
DeleteFcn	在模块被删除前执行。对 Subsystem 模块是递归的
DestroyFcn	当模块被破坏后执行
InitFcn	在模型图编译前和参数赋值前执行
LoadFcn	在模型图载入后执行。对 Subsystem 模块递归
ModelCloseFcn	在模型图被关闭前执行。对 Subsystem 模块递归
MoveFcn	当模块被移动或改变大小时执行
NameChangeFcn	在模块名或路径改变后执行。当一个 Subsystem 模块的路径被改变时，在调用它自己的 NameChangeFcn 调回例程后，对其所包含的所有模块递归地调用此函数
OpenFcn	当模块被打开时执行。这个参数通常与 Subsystem 模块一同使用。双击模块或调用作为模块自变量 open_system 命令时执行。OpenFcn 参数覆盖与打开模块相关的规定操作，显示该模块对话框或打开子系统
ParentCloseFcn	在关闭一个包含该模块子系统之前，或在使用 new_system 命令(参见第 10 章 new_system 命令)使该模块成为一个新子系统的一部分时执行
PreSaveFcn	在模型图被保存前执行。对所有 Subsystem 模块递归
PostSaveFcn	在模型图被保存之后执行。对所有 Subsystem 模块递归
StartFcn	在模型图被编译之后且仿真开始之前执行
StopFcn	任意终止仿真时执行
UndoDeleteFcn	当撤消模块删除时执行

3.11 有关模型构建的提示

本节给出一些在模型构建中可能出现的提示信息。

(1) 内存问题。一般情况下，内存越大，Simulink运行越可靠，速度越快。

(2) 使用分层结构。对于复杂的模型，最好采用分层结构，也就是增加嵌套子系统和模块分组方式。这不仅可以简化模型的顶层结构，提高模型的可读性，而且更便于他人理解。见“创建子系统”一节。另外，模型浏览器(见本章“模型浏览器”)提供了关于复杂模型很多有用信息。

(3) 整理模型。合理的组织结构和清晰的模型标注使模型易于阅读和理解。信号标签和模型注释能帮助描述模型功能等。为了避免不必要的烦恼，在构建模型初始阶段，就应留意尽可能加一些注解和标识。参见本章“信号标签”和“注释”。

(4) 建模策略。如果在建模过程中，发现经常要使用相同的模块，就可以把该模块保存在一个模型中，在建立新模型时，只需打开这个模型，复制该模块就可以了。还可以建立一个模块库，把一些常用的模块收集到一个系统中并保存此系统。这样，在MATLAB命令窗口中键人系统的名字，就可以打开此系统。

通常，构建一个模型时，首先在纸上设计，然后再上机建立。在组织模型的模块时，先将模块放置到模型窗口中，然后再添加线连接模块。这样，可减少打开模块库的次数，提高工作效率。

3.12 构建方程式模型

对于新的Simulink用户来说，在建模过程中，最不好理解和解决的问题之一就是如何给方程式建立模型。下面给出了一些例子，帮助理解方程式建模。

3.12.1 摄氏-华氏温度转换模型

将摄氏温度转换为华氏温度的方程式为：

$$T_F = 9/5(T_C) + 32$$

步骤1：首先，考虑要用到的模块：

一个Ramp(斜坡函数)模块，用于产生输人温度信号，来自输人源(Sources)库；

一个Constant(常数)模块，定义常量值32，也来自输人源(Sources)库；

一个Gain(增益)模块，将输人信号与9/5相乘，来自数学(Math)库；

一个Sum(求和)模块，将两个量相加，也来自数学(Math)库；

一个Scope(示波器)模块来显示该输出，来自接受器(Sinks)库。

步骤2：把模块加到模型窗口中，如图3.49所示。

步骤3：设置Gsin模块和Conatant模块的参数值，分别打开(双击)两个模块，输人适当的值。然后，单击Close按钮确定参数值，关闭对话框。

步骤4：连接模块(图3.50)。

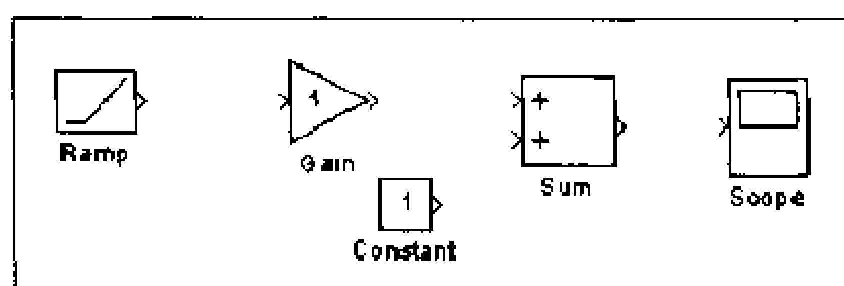


图 3.49 摄氏-华氏温度转换模型创建步骤示图 1

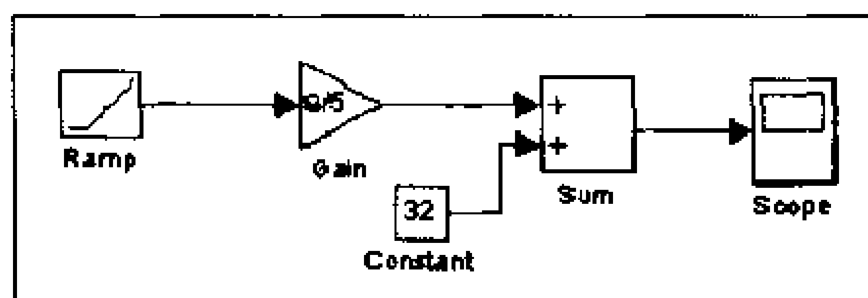


图 3.50 摄氏-华氏温度转换模型创建步骤示图 2

步骤 5：Ramp 模块输出摄氏温度。打开该模块，改变 Initial Output(初始输出)参数为 0；Gain 模块将温度与常数 9/5 相乘；Sum 模块把 32 加到相乘的结果，并输出华氏温度。

步骤 6：双击 Scope 模块用于观察输出结果。

步骤 7：仿真。从 Simulation 菜单选择 Start 项来运行仿真。该仿真将运行 10 s。

3.12.2 建立一个简单的连续系统模型

模型的微分方程为：

$$x'(t) = -2x(t) + u(t)$$

其中 $u(t)$ 是幅度为 1，频率为 1rad/s 的方波。

方波采用 Signal Generator(信号发生器)模块并选择 Square Wave，把 Units 的默认单位改为 rad/sec；Integrator(积分器)模块将它的输入积分生成 x；此模型需要的其他模块有 Gain 模块和 Sum 模块。同样，我们要用 Scope 模块看输出结果。根据上面的步骤组织模块和定义 Gain 模块参数。

方案 1：在本模型里，我们要将 Gain 模块的方向掉过来。选中该模块，然后从 Format 菜单中选择 Flip Block 命令，模块就会水平改变方向。从 Integrator 模块的输出到 Gain 模块要画分支线，在画该连线时，按下 Ctrl 键。现在将各个模块连接起来，见图 3.51。

模型中值得重视的是环路问题，该环路包含 Sum 模块、Integrator 模块和 Gain 模块。在方程式中，是 Integrator 模块的输出，同时也是计算的模块的输入，这种关系通过一个环路实现。

Scope 显示每个步进的值，由于仿真持续 10s，输出波形如图 3.52 所示。

方案 2：本例中的模型方程式还可以用一个传递函数表示。使用 Continuos 库中的 Transfer Fcn(传递函数)模块，运用拉普拉斯变换，u 为输入，x 为输出；用 sx 代替 x'，上面的方程式就成为：

$$sx = -2x + u$$

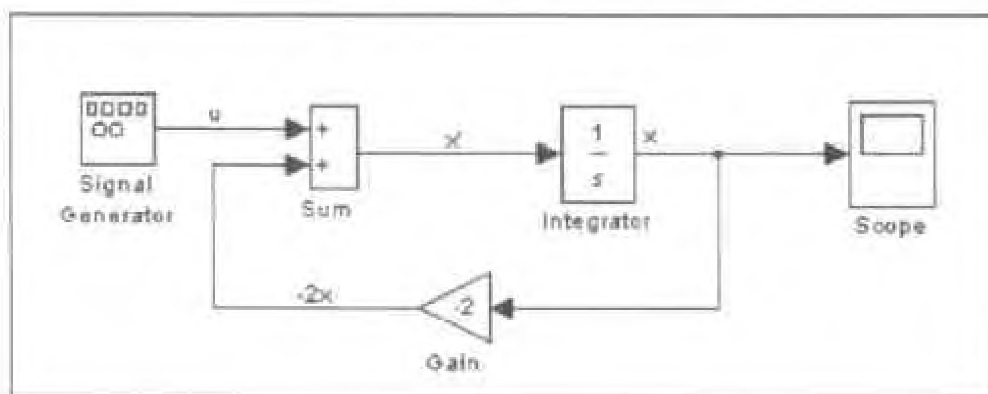


图 3.51　一个连续系统模型的方案一

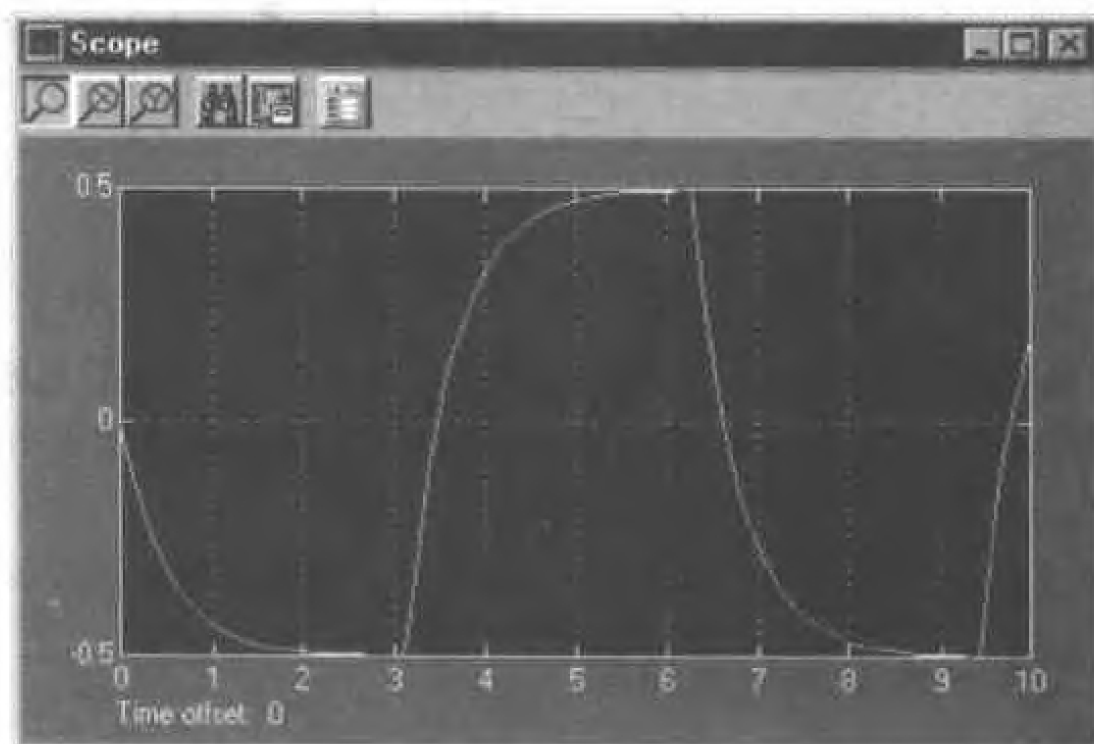


图 3.52　输出信号波形

则模型回路的传递函数为：

$$x/u = 1/(s+2)$$

Transfer Fcn 模块用参数来指定分子和分母的系数。在本例中，分子是 1，分母是 $s+2$。按 s（拉普拉斯算子）降次幂顺序定义两项的系数矢量，则分子为[1]（或 1），分母为[1　2]。这样模型就变得相当简单了，如图 3.53 所示。

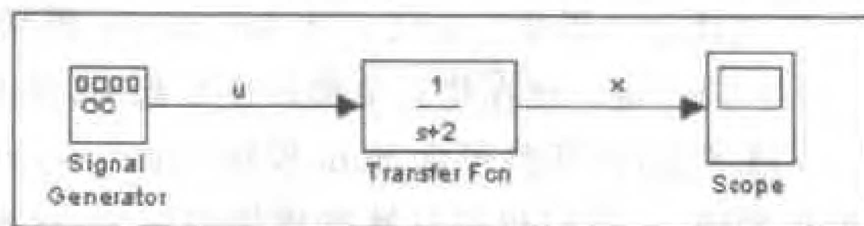


图 3.53　用 Transfer Fcn 模块定义的连续系统

该模型的仿真结果与方案一的模型仿真结果相同。

3.13　保存模型

保存一个模型，可以从 File 菜单中选择 Save(保存)或 Save As(另存为)命令。Simulink 以一种特别的格式文件(扩展名为.mdl)保存模型，文件叫做模型(model)文件，包含模型图和模块属性。模型文件的格式在附录中给出。

如果用户是第一次保存模型，使用 Save 命令。文件名必须以字母开始，可以有数字、下划线等，但总长不能超过31个字符。

如果保存模型的文件名已经存在，使用 Save 命令会替换原来的文件内容。用 Save As 命令可将模型以一个新名保存，或保存到新位置。

在确定保存模型后，Simulink 就执行下列步骤：

(1) 如果模型的 mdl 文件已经存在，它被重新命名为一个临时文件。

(2) Simulink 执行所有模块的 PreSaveFcn 调回例程，然后执行该模型图的 PreSaveFcn 调回例程。

(3) Simulink 将模型文件写到使用同一名字和 mdl 扩展名的一个新文件中。

(4) Simulink 执行所有模块的 PostSaveFcn 调回例程，然后执行模型图的 PostSaveFcn 调回例程。

(5) Simulink 删除临时文件。

如果在这个过程期间有错误发生，Simulink 把临时文件名改为原来模型文件的名字，将当前版本的文件写入带有扩展名.err 的文件中，并且提示出错信息。即使错误发生较早，Simulink 仍执行步骤(2)到(4)。

3.14　打印模型图

打印模型图，可以通过 File 菜单中选择 Print 项(在 WINDOWS 系统)，或者在 MATLAB 命令窗口中输入 print 命令(对所有的平台)。

在 Windows 系统，Print 菜单项打印当前窗口的模型图。

3.14.1　打印对话框

在选择 Print 后，会出现一个 Print 对话框。该对话框让用户有选择地打印模型中的系统。通过对话框，可以完成下列任务：

(1) 只打印当前系统；

(2) 打印当前系统和其分层结构以上的所有系统；

(3) 打印当前系统和其分层结构以下的所有系统，并有选择项，可以打印封装模块和库模块的内部内容；

(4) 打印模型中所有的系统，有选择地打印封装模块和库模块的内部内容；

(5) 打印每张图上的外框结构。

在各个操作系统平台上，Print 对话框提供的可选择的打印功能是基本相同的。

例1. 图3.54显示了 Windows 系统下，Print 对话框的内容，该图只打印当前系统。

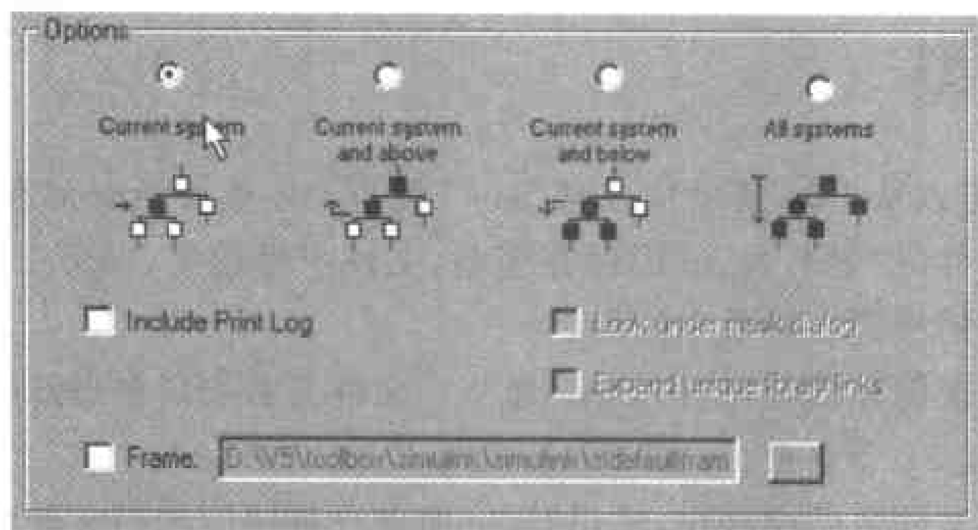


图 3.54　打印对话框(选择打印"当前系统")

例 2. 当用户选择 Current system and below，或 All systems 的选项后，复选框就被激活。在图 3.55 所示的这张图里，选择了 All systems，即所有的系统。

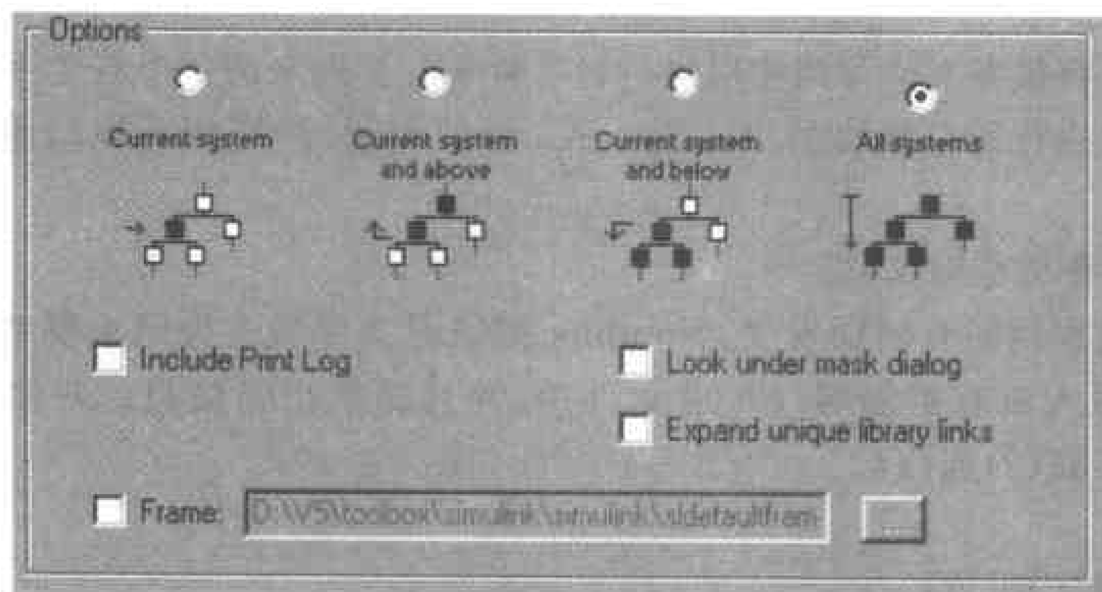


图 3.55　打印对话框(选择打印"全部系统")

例 3. 选中 Look Under Mask Dialog 复选框，在当前模块层次或当前模块层次以下，遇见封装的子系统时，就会打印子系统的内容。当打印所有系统时，最高层次的系统被认为是当前模块层次，因此，Simulink 打印所有的封装模块的内容。

例 4. 选中 Expand Unique Library Links 复选框，当模块为系统时，会打印库模块的内容。即使模型中有多个复制模块，也只打印一个。

例 5. 选中 Include Print Log 复选框，打印记录。打印记录列出了打印的模块和系统。

例 6. 选中 Frame 复选框，在每个图中只打印模块结构框架，并要在旁边的编辑框内输入该框架文件的路径。用户可以使用 MATLAB 的外框编辑器制定自己的模块框架。有关使用外框编辑器的信息可以参看 frameedit 命令的帮助。

3.14.2　使用命令打印

1. 命令格式

```
print -ssys -device filename
```

sys 是要打印的系统名。在系统名的前面，必须加一个切换标识符 S。当然 sys 必须是打

开的，或在当前操作过程中已经打开。如果在系统名中包含了空格或超过一行，必须将系统名指定为字符串（见举例）。

device 指定设备类型。关于设备类型的列表和描述，参看 MATLAB 有关图形打印的说明。

filename 是保存输出的 PostScript 文件。如果 filename 已存在，将会被替换。如果 filename 没有扩展名，会自动加上适当的扩展名。

2. 举例

例 1. 下面的命令打印一个名为 untitled 的系统：

```
>>print -suntitled
```

例 2. 打印当前系统中名为 Sub1 的子系统的内容：

```
>>print -sSub1
```

例 3. 打印名为 Requisite Friction 的子系统的内容：

```
>>print ([ '- sRequisite Friction'])
```

例 4. 打印一个名为 Friction Model 的系统。因为名字中出现空格，所以第一个命令指定换行符为一个变量，第二个命令打印系统。

```
>>cr = sprintf(' n');
>>print ([ '- sFriction' cr 'Model'])
```

3.14.3 指定纸张大小和打印方向

Simulink 让用户指定打印模型图表所用的页面类型和打印方向，通过使用 set_param 命令分别设置模型的 PaperType 和 Paper Orientation 属性（见附录“模型参数”一节）。也可以使用 MATLAB 的 orient 命令，只设置打印方向。在 Windows 下，用户可以通过 Print 对话框设置页面类型和打印方向。

3.14.4 定位和调整图表大小

可以使用模型的 PaperPositionMode 和 PaperPosition 参数来对图表进行定位和调整尺寸。PaperPosition 参数的值是一个窗体矢量[left bottom width height]([左 底 宽度 高度])。前两个元素指定一个矩形在页面的左下角起始位置，后两个元素指定矩形的宽度和高度。当模型的 PaperPositionMode 是 manual(手动)模式时，如果有必要，Simulink 会自动调整比例并定位模型图表，使之适合指定要打印的矩形。例如在执行下面的命令后，就在一张美国信纸页(37.78cm×27.94cm)的左下略低处以横向打印 vdp 模型图。

```
>>vdp
>>set_param(' vdp', 'PaperType', 'usletter')
>>set_param('vdp', 'PaperOrientation', 'landscape')
>>set_param(' vdp', 'PaperPositionMode', 'manual')
>>set_param(' vdp', 'PaperPosition', [0.5 0.5 4 4])
>>print - svdp
```

如果 PaperPositionMode 是自动模式，Simulink 会将模型图置于页面的中心，如果有必要，还将自动调整图的比例。

3.15　模型浏览器

模型浏览器的用途主要有：

(1) 有层次地定位一个模型。

(2) 直接在一个模型中打开系统。

(3) 确定模型中包含的模块。

编者提示：模型浏览器在 Windows 系统和 UNIX 系统中是不同的。

3.15.1　Windows 系统的模型浏览器

从 View 菜单中选择 Model Browser，显示模型浏览器面板。当前模型的窗口将分裂为两个窗，左窗是浏览器，显示的是系统的树状结构；右窗显示的是相应的系统或子系统模型图。见图 3.56 所示的例子。

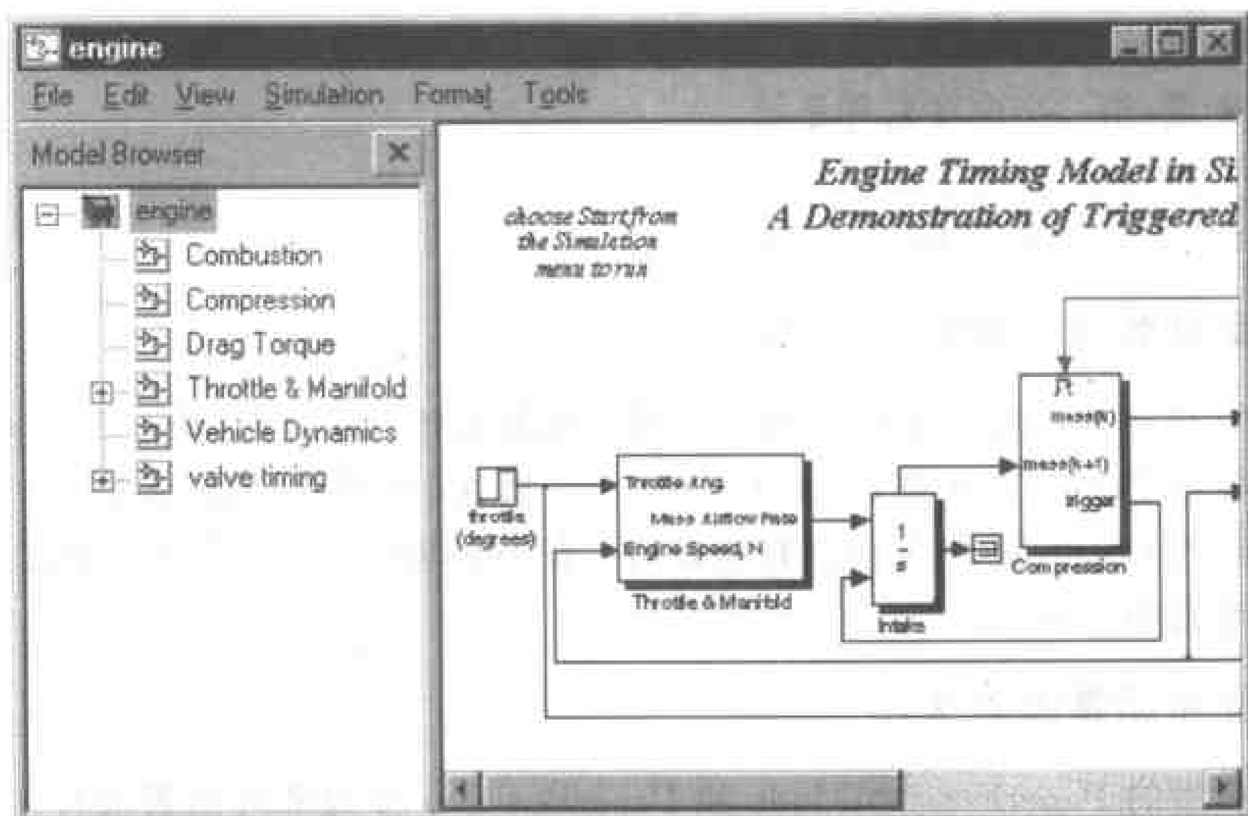


图 3.56　Windows 下模型浏览器

树状结构中的每一项对应模型中的一个子系统。单击各个子系统旁边的节点[+]或[−]框可以展开或折叠树，也可以按←、→、↑或↓箭头键，移动选项。

3.15.2　使用 UNIX 系统模型浏览器

打开模型浏览器，从 File 菜单中选择 Show Browser(显示浏览器)，浏览器窗口出现，显示当前模型的信息。图 3.57 为浏览器显示 clutch 系统的内容。

1. 浏览器窗口的内容

(1) 系统列表。左窗中的列表为当前系统和所包含的子系统。

(2) 模块列表。右窗中的列表为所选系统所包含的模块名。初始情况下，显示顶层系统的模块。

(3) File 菜单，它包含 Print、Close Model 和 Close Browser 菜单项。

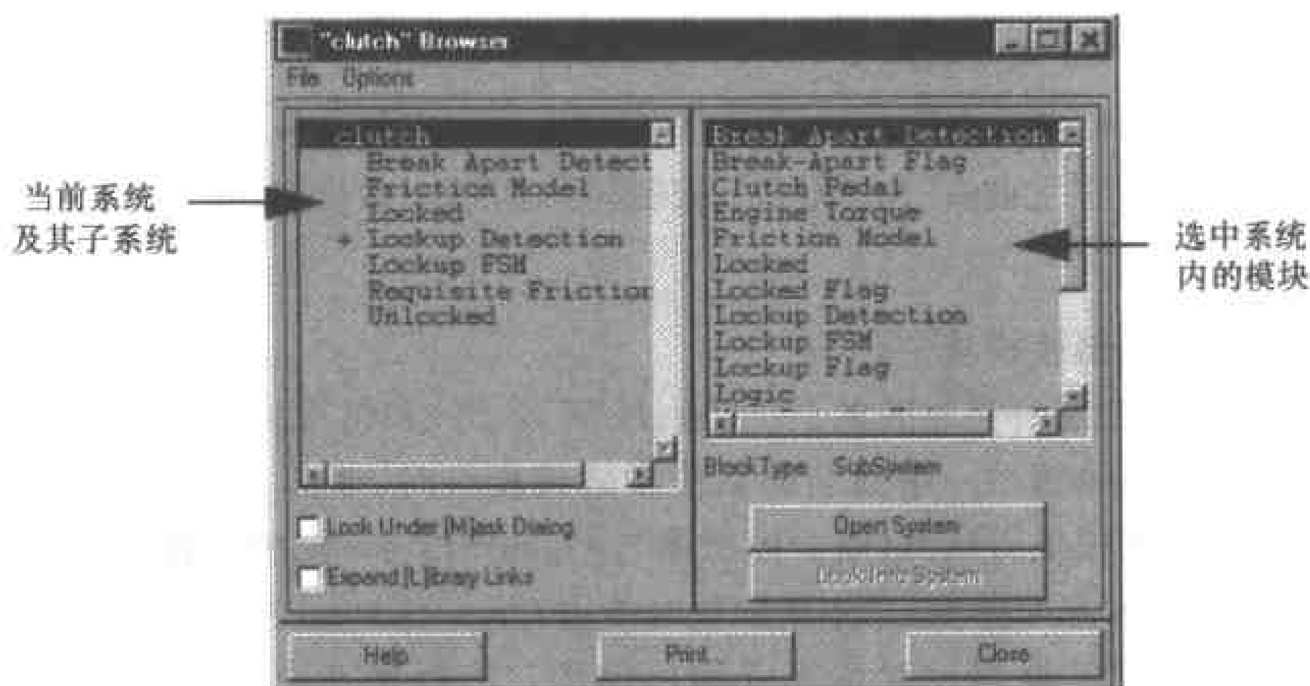


图 3.57 UNIX 下模型浏览器

(4) Option 菜单，它包含这些菜单项：Open System(打开系统)，Look Into System(查看系统)，DisplayAlphabetical/Hierarchical List(显示按字母/层次列表)，Expand All(展开所有)，Look Under Mask Dialog(查看 Under Mask 对话框)，和 Expand Library Links(展开所有库链接)。

(5) Option 复选框和按钮：Look Under [M]ask Dialog 和 Expand [L]ibrary Links 复选框，及 Open System 和 Look Into System 按钮。默认时，并不显示封装模块内容和库链接的模块。这些复选框的选择可以改变默认值。

(6) 所选模块类型。

(7) 对话框按钮：Help(帮助)，Print(打印)，Close(关闭)。

2. 解释列表内容

Simulink 识别封装模块、参考模块，定义 OpenFcn 参数的模块和包含子系统的系统，在模块或者系统名字之前使用下面这些符号以示特征：

(1) 系统名字之前的加号(+)，表示该系统是可扩展的，即该系统之下还包含系统。在模块列表中，双击系统名字，可以展开列表和显示它的内容。当一个系统展开时，(—)号出现在名字之前。

(2) [M]表示该模块是封装的，有一个封装对话框或封装工作空间。关于封装的更多信息，请参见第 6 章。

(3) [L]表示该模块是参考模块。更多信息，参看本章“库”。

(4) [O]表示该模块定义了一个打开函数(OpenFcn)调回。关于模块收回的更多信息，请参见本章“使用调回例程”。

(5) [S]指示该系统是一个 Stateflow 模块。

3. 打开一个系统

用户可以打开在模块列表中列出的任意系统或模块。

(1) 在系统列表中，通过单击选择包含需要打开的系统的上级系统名字，上级系统的内容会出现在模块列表上。

(2) 依据系统是否封装，是否与库模块关联，或定义有打开函数调回，用户可以按下列操作打开系统：

①如果系统左边没有特征符号，双击它的名字，或者选中它的名字，单击 Open System 按钮。

②如果在该系统名字之前有[M]或[O]特征符号，则选中系统名，单击 Look Into System 按钮。

4. 查看封装系统或链接模块

默认情况下，模型浏览器把封装的系统(通过[M]识别)和链接模块(通过[L]识别)看作模块而不是子系统。如果选中一个封装的系统或链接模块时，单击 Open System 按钮，模型浏览器将显示该系统或模块对话框(双击模型图与 Open System 按钮功能一样)。类似地，如果模块定义了 OpenFcn 调回参数，选中该模块时单击 Open System 按钮，则执行调回例程。

用户可以使模型浏览器越过对话框或调回例程。选择列表中的模块，单击 Look Into System，模型浏览器将显示系统或模块的内容。

5. 按字母顺序显示列表内容

默认时，该系统列表按分层结构显示模型系统，包含在系统名前的一个特征符号(—)。当这些系统展开时，模型浏览器在系统的名字之前显示一个(—)。要按字母顺序显示系统，选择 Option 菜单中的 Display Alphabetical List 项。

3.16 追踪模型版本

一个 Simulink 模型在开发过程中要经历许多次修改。为帮助用户跟踪变化的版本，Simulink 会生成和存储版本控制信息，包括版本号，创建和最近一次更新该模型的人名，以及其他变化的记录。下面介绍的 Simulink 特性有助于管理和使用版本控制信息：

(1) Model Parameters(模型参数)对话框可以让用户编辑储存在模型中的一些版本控制信息，和选择不同的版本控制内容。

(2) Model Info(模型信息)模块，可以显示版本控制信息，包括通过外部的版本控制系统维护信息，在模型图表作为解释模块。

(3) Simulink 版本控制参数允许用户从 MATLAB 命令行或 M 文件中存取版本控制信息。

3.16.1 指定当前用户

当用户创建或更新一个模型时，Simulink 将用户的名字写在模型中，以便于模型版本控制。Simulink 假定用户名至少是用下列环境变量中的一个来指定的：USER，USERNAME，LOGIN 或 LOGNAME。如果用户的系统没有定义任意一个这样的变量，Simulink 就不会更新模型的用户名。

UNIX 系统总是定义 USER 这个环境变量并将其设置到用户用于登录系统的用户名中。因此，如果用户使用的是 UNIX 系统，就没有必要让 Simulink 把你作为当前用户。另一方面，对 Windows 系统，可以定义一些或者不定义任意用户名环境变量，这依赖于使用的 Windows 系统的版本，以及它是脱机运行还是连接到网络。可以使用 MATLAB 的 getenv 命令来确定是否已经定义过环境变量。例如，在 MATLAB 命令窗口输入命令：

```
>>getenv('user')
```

来确定用户的 Windows 系统是否已经定义了用户环境变量。如果没有,必须自己定义。在 Windows 95/98 系统下,在系统的 autoexec.bat 文件中输入命令:

```
>>set user = yourname
```

其中 yourname 是在模型文件中的识别名字。然后,重新启动计算机。

编者提示:autoexec.bat 文件在系统硬盘的 C:\目录下。

在 Windows NT 下,使用系统属性对话框中环境变量面板指定 USER 环境变量。见图 3.58 所示。

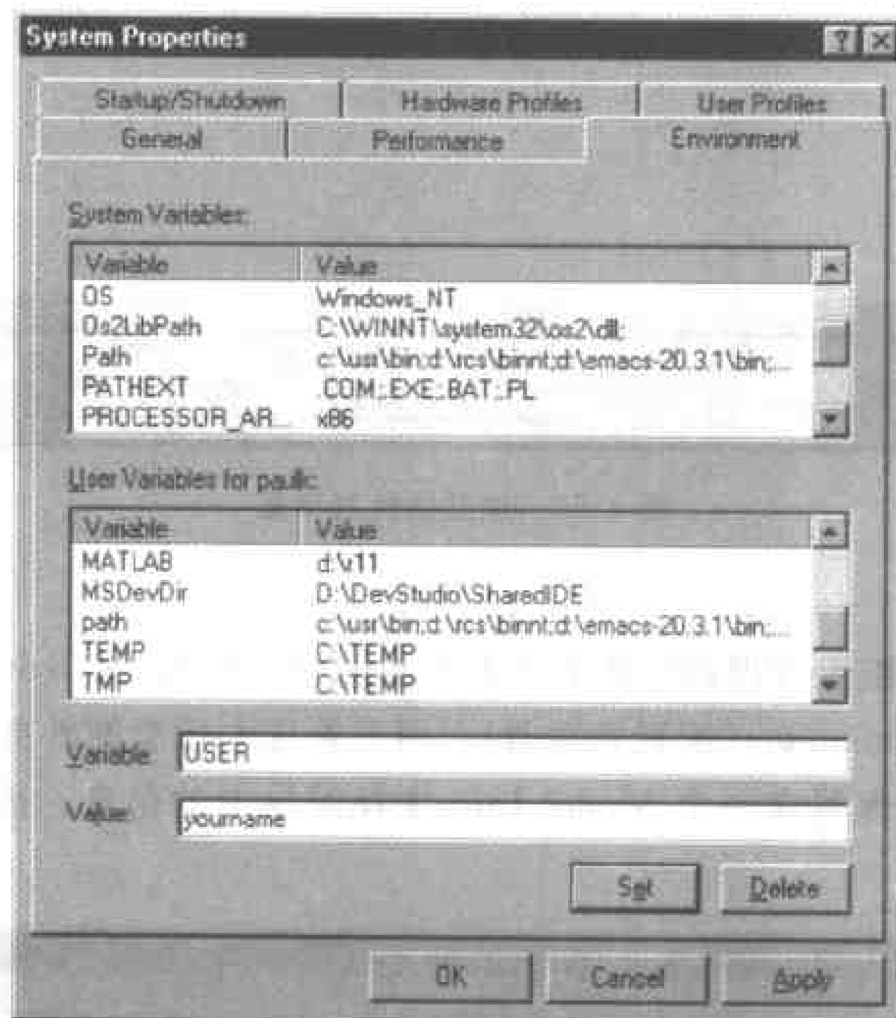


图 3.58 WINDOWS NT 下使用系统属性指定 USER 环境变量

显示系统属性对话框,在我的电脑文件夹中选择系统控制面板文件夹中的系统。指定用户变量,在变量域中输入 USER,在值域中输入用户的登录名,并且选择建立连接(Set)按钮。然后选择 OK 关闭对话框。

3.16.2 模型属性对话框

模型属性对话框允许用户编辑一些版本控制参数和设定一些相关项。从 Simulink 的 File 菜单中选择 Model Parameters 来显示该对话框,如图 3.59 所示。

1. Model Properties 选项

用于编辑下列的版本控制参数:

(1) Creator(创建者)。创建本模型的用户名。当创建该模型时,Simulink 将本属性的值设置为 USER 环境变量的值,编辑本栏来改变该值。

(2) Created(创建)。创建本模型的日期和时间。

(3) Model description(模型描述)。模型的描述。

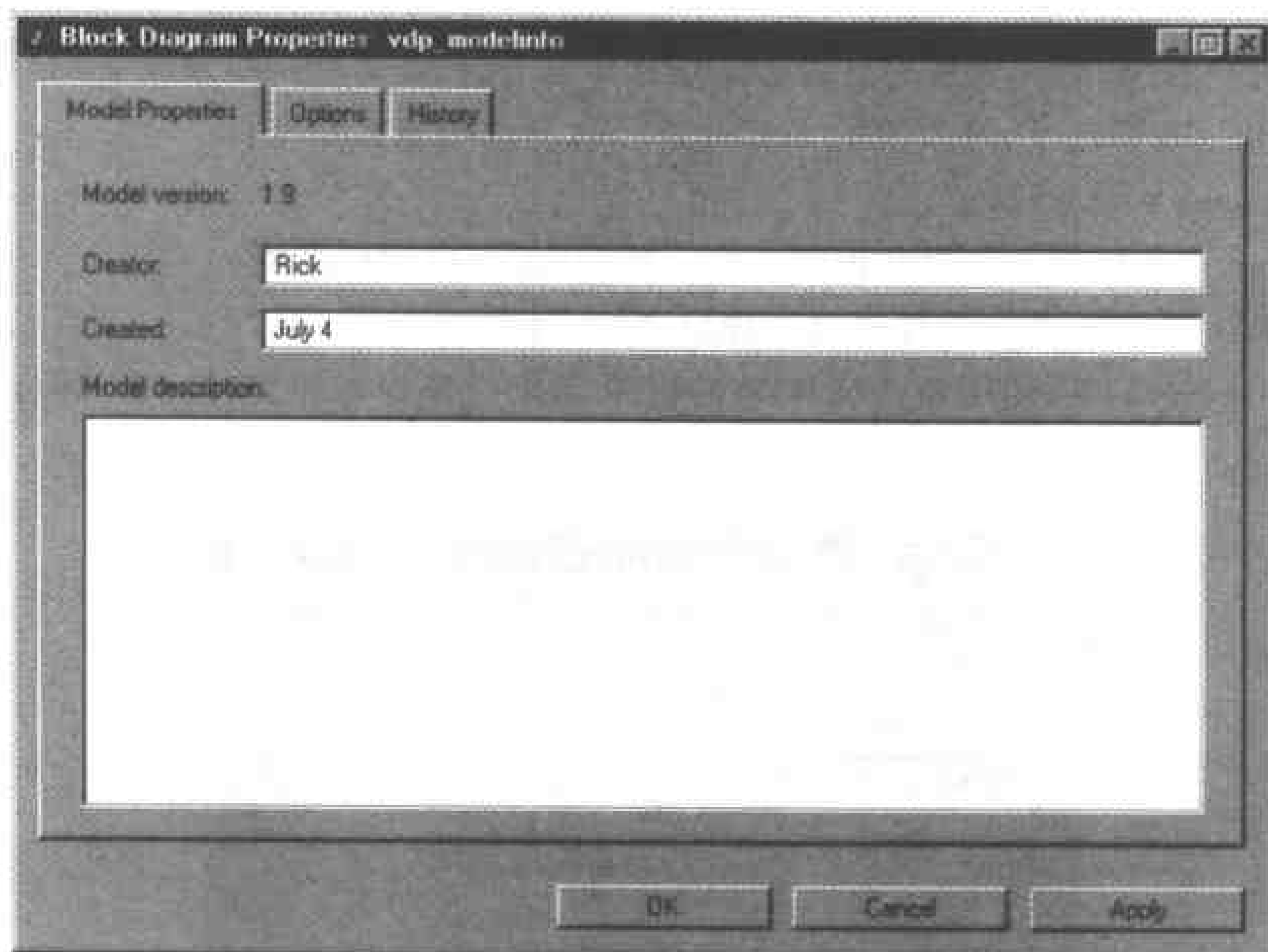


图 3.59　模型属性对话框

2. Option 选项

用户可以选择一个配置管理器并指定版本控制信息的格式(图 3.60)。

(1) Configuration manager(配置管理器)。用于管理本模型的外部配置管理器。选择本项可以包含来自配置管理器在一个 Model Info 注释模块中的信息。欲知更多信息详见第 7 章"Model Info 模块"。

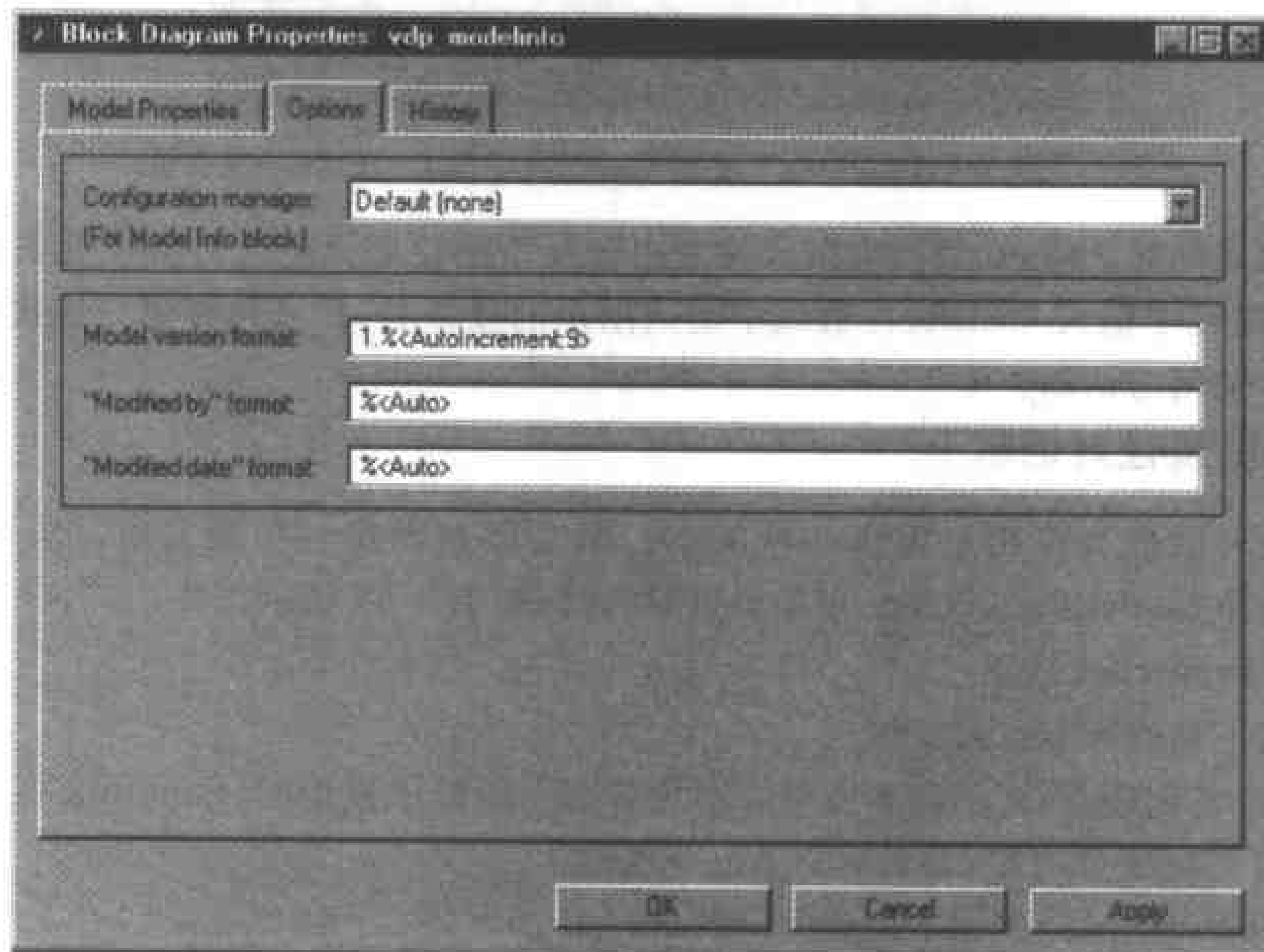


图 3.60　模型选项对话框

在 MATLAB 的/toolbox/local 目录下的 cmopts. m 文件为模型指定了默认的配置管理器。默认值(default)为 none 表示没有指定任何配置管理器，用户可以编辑该文件来指定一个配置管理器。

(2) Model version format(模型版本格式)。在模型参数面板和模型信息模块中显示模型版本号的格式。本参数的值可以是任意文本字符串，可以包含标记%<AutoIncrement:#>，其中#是一个整数。当显示模型版本号时，Simulink 用#替换该标记。例如，将

```
1.%<AutoIncrement:2>
```

显示为 1.2。

当保存该模型时，#增加 1。例如当保存模型时，

```
1.%<AutoIncrement:2>
```

变成

```
1.%<AutoIncrement:3>。
```

(3) Modified by format。用于显示 History 窗格、历史记录和 Model Info 模块中"Modified by"值的格式。此域的值可以是任意字符串，可以包含标记%<Auto>。Simulink 用当前 USER 环境变量的值替换本标记。

(4) Modified date format(改变日期格式)。用于显示 History 面板、历史记录和 Model Info 模块中的"Last modified date"值的格式。此域的值可以是任意字符串，可以包含标记%<Auto>。Simulink 用当前日期和时间替换本标记。

3. History 选项

用户可以通过次选项激活、观看、编辑模型的变化历史，参见图 3.61。

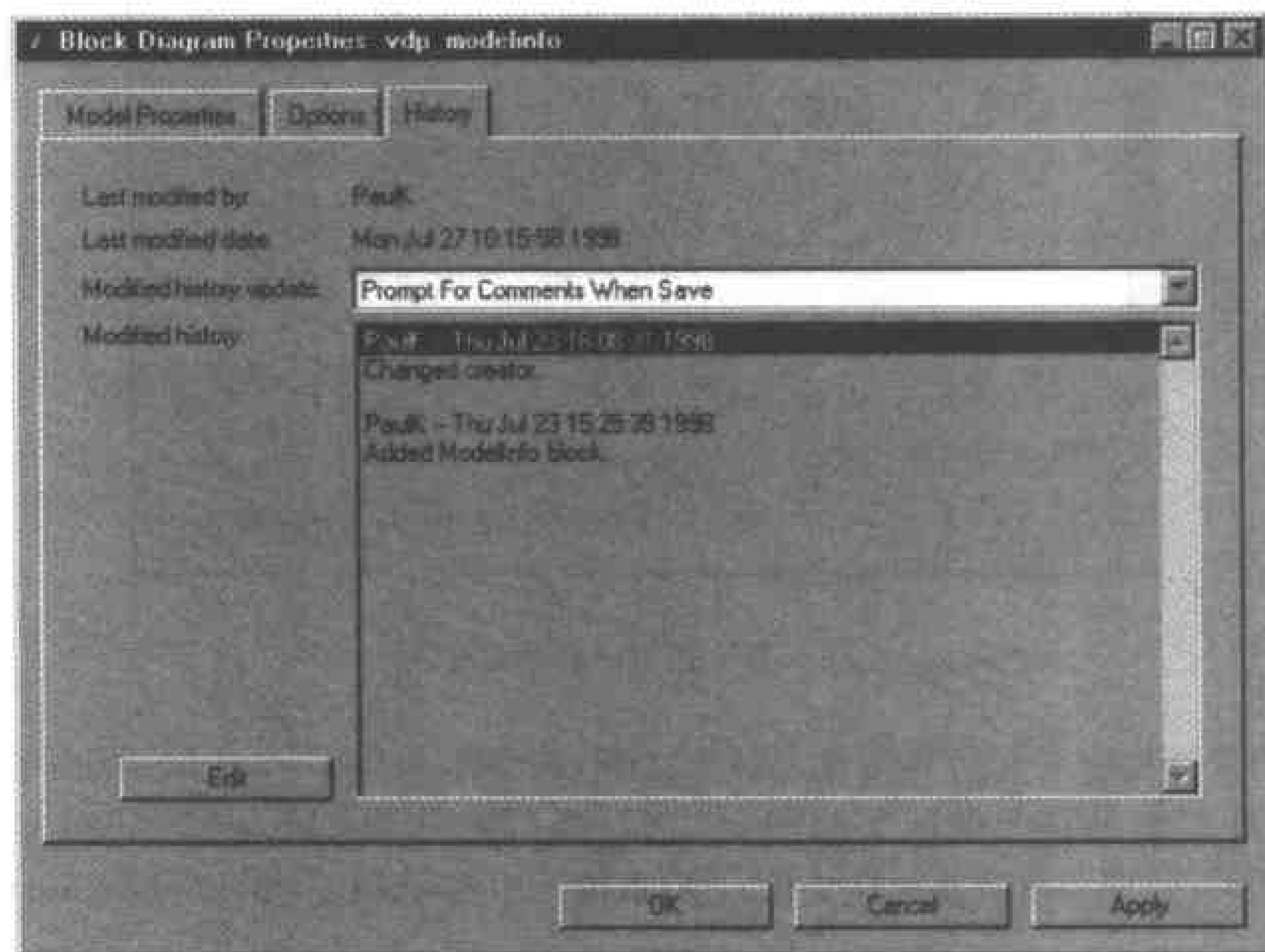


图 3.61　模型历史对话框

(1) Last modified by。最近一次修改本模型的人名。当保存该模型时，Simulink将本参数的值设置为USER环境变量的值。用户不能编辑本域中的内容。

(2) Last modified date。最近一次修改本模型的日期。Simulink将本参数的值设置为保存模型时的系统日期和时间。用户不能编辑本域中的内容。

(3) Modified history update。指定当保存该模型时，是否提示用户一个注释信息。如果选择了"Prompt for Comments When Save"，当保存模型时，将会弹出一个注释框，供输入要保存的模块的信息。在注释栏，你可以记录当前模型的任意改变。Simulink将此参数的以前的值保存在模型变化历史中。更多信息请看下面"创建模型的变化历史记录"。

(4) Modified history。本模型的修改历史记录。当更新模型时，Simulink编译由用户输入的注释信息。通过选择旁边的Edit按钮，用户可以在任意时候编辑该历史记录。

3.16.3　创建模型的变化历史记录

Simulink允许用户在模型本身内部创建和存储一个模型变化的历史记录。当保存模型时，Simulink自动编译由用户输入的变化注释。

1. 记录变化

开始一个变化历史记录，在Model Properties对话框的History窗格上选中"Prompt for Comments When Save"项。在下一次保存该模型时，Simulink显示一个Log Change对话框，如图3.62所示。

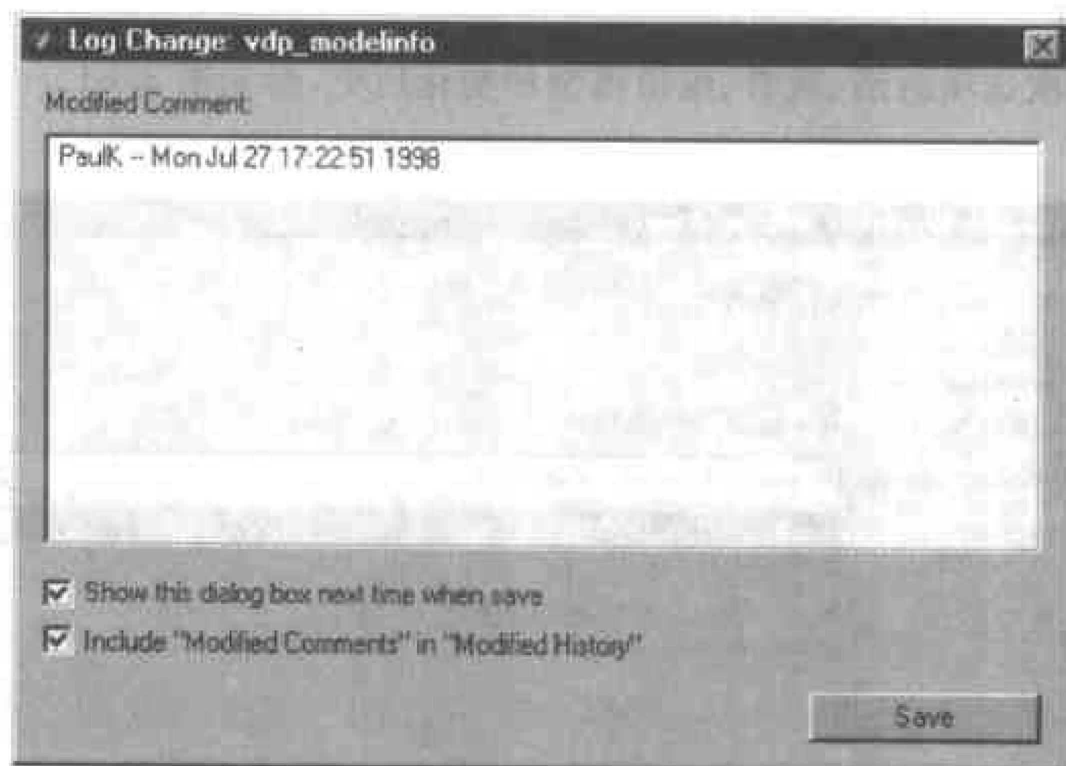


图3.62　历史记录对话框

如果用户要在模型变化历史记录上加入注释，在Modified Comments域内输入注释，并按Save键保存。如果在此期间不想输入注释，取消选中Include "Modified Contents" in "Modified History"项。如果要放弃变化记录，取消选中Show this dialog box next time when save项。

2. 编辑变化历史记录

编辑一个模型的历史记录，单击Model Properties对话框中的Edit按钮，将会在一个

Modification History 对话框显示模型的历史记录，如图 3.63 所示。

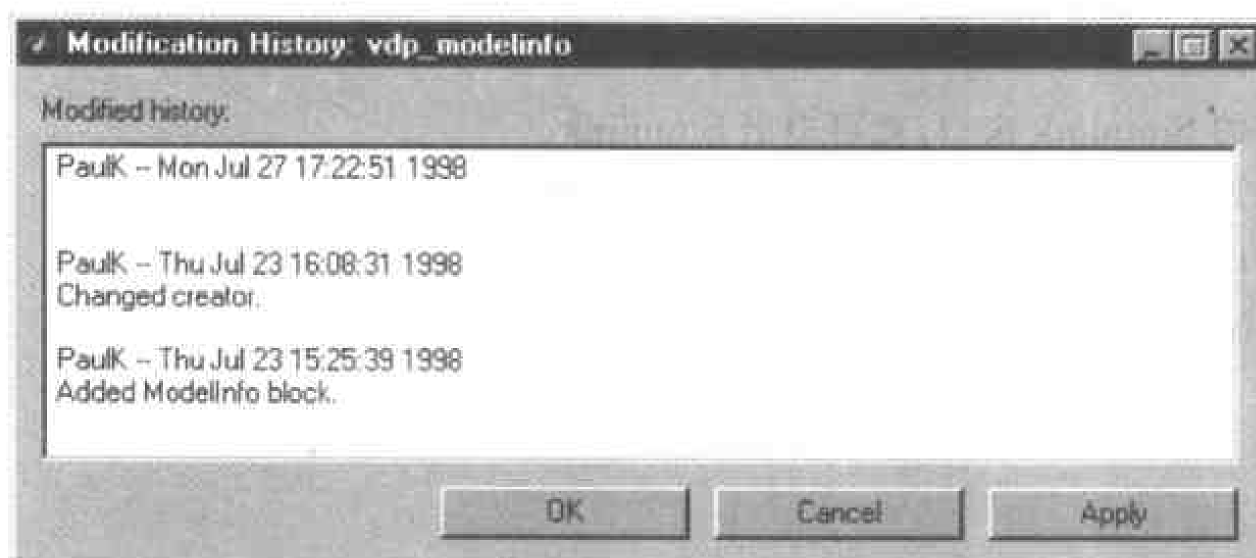


图 3.63 编辑历史记录对话框

编辑该对话框中的历史，并选择 Apply 或 OK 保存该变化。

3.16.4 版本控制属性

Simulink 将版本控制信息作为模型参数存储在模型里。用户可以使用 Simulink 的 get_param 命令，从 MATLAB 命令窗口或从 M 文件中访问本信息。表 3.11 给出了用来存储 Simulink 版本控制信息的模型参数。

表 3.11 用于版本控制信息的模型参数

属　性	描　述
Created	创建日期
Creator	创建者的名字
ModifiedBy	上次修改本模型的人
ModifiedByFormat	ModifiedBy 参数的格式，值可以是字符串，字符串可以包含标记%<Auto>。Simulink 用 USER 环境变量的当前值替换该标记
ModifiedDate	修改日期
ModifiedDateFormat	修改日期格式，值可以是字符串，字符串可以包含标记%<Auto>。当保存模型时，Simulink 用当前日期和时间替换该标记
ModifiedComment	由最近一次更新模块的用户输入的注释
ModifiedHistory	模型的变化历史记录
ModelVersion	版本号
ModelVersionFormat	模型版本号的格式，可以是字符串，字符串可以包含标记%<AutoIncrement:#>，其中#是一整数，当显示版本号时，Simulink 以#替换该标记。每次保存模型时，#值增加 1
Description	模型描述
LastModificationDate	上次修改日期

3.17 退出 Simulink

关闭所有的 Simulink 窗口，就可退出 Simulink。

退出 MATLAB，可从 File 菜单中选择下面其中一个命令：

(1) 在 Windows 系统：Exit MATLAB；

(2) 在 UNIX 系统：Quit MATLAB。

第4章 Simulink 仿真模型动态调试器

在构建模型的过程中，出现这样那样的错误是不可避免的。作为模型设计人员，最需要知道错误发生的地方以及原因。Simulink 给我们提供了这样的工具——Simulink 仿真模型动态调试器，它可以帮助我们提高查寻和修正错误的效率。Simulink 仿真模型动态调试器是一种能自动寻找和诊断 Simulink 仿真模型中错误的工具。用户可以通过调试器提供的顺序运行单步仿真和显示中间模块的状态、输入和输出等功能来迅速查找问题所在。

主要内容：介绍如何使用调试器来诊断 Simulink 模型，主要包括：调试器的使用，运行参数设置，错误信息说明以及调试命令。

学习目的：使掌握了前两章、并建立了自己的仿真模型的读者，学习掌握如何检查模型错误的动态调试方法。

4.1 调试器使用概述

4.1.1 启动调试器

在 MATLAB 命令窗口输入 sldebug 命令或 sim 命令(option 需设置为调试选项)就可启动在调试器控制下的一个模型。使用 sim 命令句法见第 5 章“sim 命令”。

我们以上一章建立的一个连续系统模型 Consys 为例。如输入下面的命令：

```
>>sim('consys',[0,10],simset('debug','on'))
```

或命令：

```
>>sldebug 'consys'
```

编者提示：sldebug 后面必须跟空格。

将上章的举例模型 Consys 装入存储器中，并将第一时间步停在第一个模块上。调试器高亮显示模型图中模型的初始模块和相关的输出信号连线。图 4.1 是在调试模式启动后的 Consys 模型图。

调试器也将仿真起始时间和调试命令提示符显示在 MATLAB 命令窗口上。命令提示符显示模块指数(见 4.1.4“模块指数”)和第一个将要运行的模块名。例如，前例中的命令在 MATLAB 命令窗口中产生下列输出：

```
[Tm = 0          ] * * Start * * of system 'Consys' outputs
(sldebug @0:0 'Consys/Integrator'):
```

在提示符后面输入不同的调试命令，我们可以获得诸如帮助，步进运行仿真，检查数据等服务；或进入调试器执行其他调试任务并在调试提示符下执行其他 MATLAB 命令。在以后

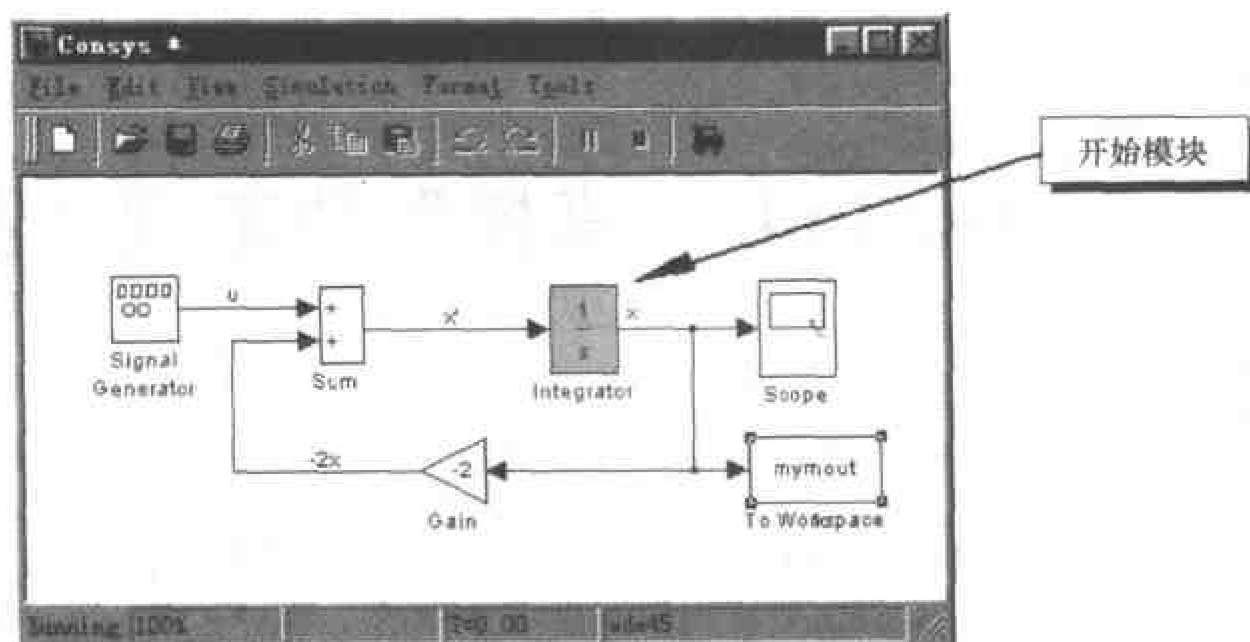


图 4.1　调试模式下的 Consys 模型图

各节中，将详细介绍如何使用调试器命令。

4.1.2　获得帮助

在调试提示符下键入 help(如同在 MATLAB 命令窗口一样)命令可以获得调试器命令的简要说明。本章最后一节将给出各个命令的详细说明。下面各节说明如何使用这些命令调试一个模型。

4.1.3　键入命令

调试器接受缩写的调试器命令。用户可以通过在 MATLAB 命令行键入一个空命令(如按下 Return 键)重复某命令。参看本章“调试器命令”中命令缩写与可重复命令表。

4.1.4　模块指数

很多 Simulink 调试器命令和信息使用模块指数以便查找模块。模块指数的表示形式为 s:b，其中 s 为标识正在调试的模型中某个系统的一个整数；而 b 为标识这个系统中某模块的一个整数。例如，模块指数 0:1 对应模型的 0 系统中模块 1。slist 命令显示正在调试的模型中一个模块的模块指数(参看本章“slist 命令”)。

4.1.5　访问 MATLAB 工作空间

用户可以在 sldebug 提示符下键入任意 MATLAB 表达式。例如，假设处于程序的干预断点，用户正在将用户模型的时间和输出分别存入 tout 和 mymout 中。则下面的命令：

```
(sldebug ...) >>whos
```

在 MATLAB 命令窗口就显示出在当前步运行后的变量及属性信息：

```
Name        Size        Bytes Class
  mymout      1x1         658 struct array
  tout        0x1           0 double array
Grand total is 24 elements using 658 bytes
(sldebug @0:0 'Consys/Integrator');
```

假设用户想访问一个名字与 sldebug 命令完全或部分相同的变量(如“s”为 step 命令的一部分)。在 sldebug 提示符下键入“s”会使模型执行一步。但是

```
(sldebug...) eval('s')
```

显示变量 s 的值。

4.2　步进运行仿真

Simulink 调试器允许用户逐步运行一个仿真。用户可以从模块到模块、时间点到时间点或断点到断点逐步运行(参见本章“设置断点”)。用户通过输入恰当的调试器命令(见表 4.1)选择运行仿真的步进量。

表 4.1　步进运行命令及功能

命令	使仿真前进……
step	一个模块
next	一个时间步
continue	到下一个断点
run	忽略所有断点,到仿真尾部

4.2.1　模块步进

在调试器提示符下,输入 step 可使仿真前进到下一个模块。调试器执行当前模块,停止,按照模型的模块执行顺序高亮显示下一个模块(见本章“显示一个模型的模块执行顺序”)。例如,输入 step 命令后,模型运行的情况如图 4.2 所示。

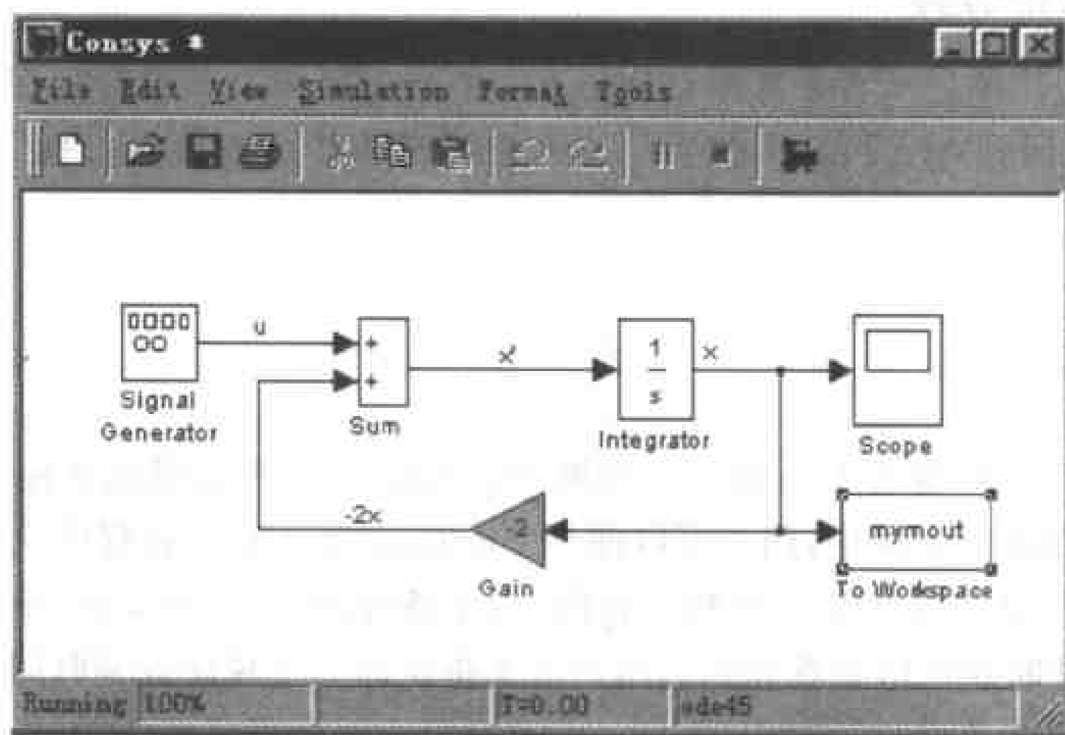


图 4.2　执行 step 命令后,模块的步进运行情况

若下一个将要执行的模块是一个子系统模块,则调试器打开子系统模块图,并高亮显示下一个模块。

在执行一个模块后，调试器输出模块的输入(U)和输出(Y)，并在 MATLAB 命令窗口重新显示调试命令提示符。调试器提示显示下一个需赋值的模块。

```
(sldebug @0:0 'Consys/Integrator'): step
U1 = [0]
Y1 = [0] (sldebug @0:1 'vdp/Out1'):
```

1. 时间步超界

当用户通过模型分序运行最后一个模块后，调试器将使仿真进入到下一个时间步，并暂停在下一个时间步中要执行的第一个模块上。为表明用户已经越过一个时间步边界，调试器将在 MATLAB 命令窗口中显示当前时间。

2. 小时间步步进

用户可进行小时间步的模块步进，也可是大时间步的步进。在调试器的命令提示符后键入 minor 为小时间步步进，再输入一次，取消。

4.2.2　时间步步进

next 命令执行在当前时间步余下的模块。本质上，该命令使用户能够用一个命令使仿真进到下一个时间步。当用户知道在余下的时间步内没有什么可感兴趣的事件发生时用此命令可以提高效率。当仿真进入下一个时间步后，调试器将在模型分序运行中的第一个模块上中断。

4.2.3　断点步进

continue 命令使仿真从当前断点进入下一个断点(详见“设置断点”)或到达仿真结束，两者取先到为有效。

4.2.4　不间断运行仿真

run 命令允许用户跳过所有干预断点从仿真中的当前断点开始运行程序直至结束。在仿真结束后，调试器返回到 MATLAB 命令行。为继续调试某个模型，用户必须再次启动调试器。

4.3　设置断点

有时，为了调试程序，需要人为设置一些断点。Simulink 调试器允许用户按照需要给仿真程序定义断点。在运行过程中，用户可以使用 continue 命令使仿真程序从一个断点运行到下一个断点。调试器允许用户定义的断点有两类：无条件断点(unconditional)和条件(conditional)断点。无条件断点在仿真首先到达用户事先指定的一个模块或是时间步时出现。条件断点在用户事先指定的条件满足时出现。

当用户分析问题所在或当一定的条件产生时，可以通过定义一个恰当的断点，并使用 continue 命令运行仿真程序。这样，程序能立刻跳到问题的出现点上。

用户也可输入适当的断点命令(见表 4.2)来设置特殊种类的断点。

表 4.2　断点命令和用途

命　令	使仿真程序中断……
break <gcb | s:b>	在一个模块的开始
bafter <gcb | s:b>	在一个模块的结尾
tbreak [t]	在一个仿真时间步
nanbreak	在下溢或溢出(NaN)或无穷大(Inf)值产生处
xbreak	当仿真到达确定仿真时间步长的状态时
zcbreak	当在仿真时间步之间产生过零时

4.3.1　模块断点

调试器允许用户在模块的起点或结尾处指定一个断点。

1. 在模块起始处设置断点

break 命令让用户在一个模块的起始处设置一个断点。在模块的起始处设置一个断点将导致调试器在达到某个时间步上的模块后中断仿真。

用户可以指定一个这样的模块，通过该模块的指数或图表设置断点。

格式一：通过图表指定模块，请务必先在模型模块图上选定该模块，并输入 break gcb 命令，执行后的情况如图 4.3 所示。

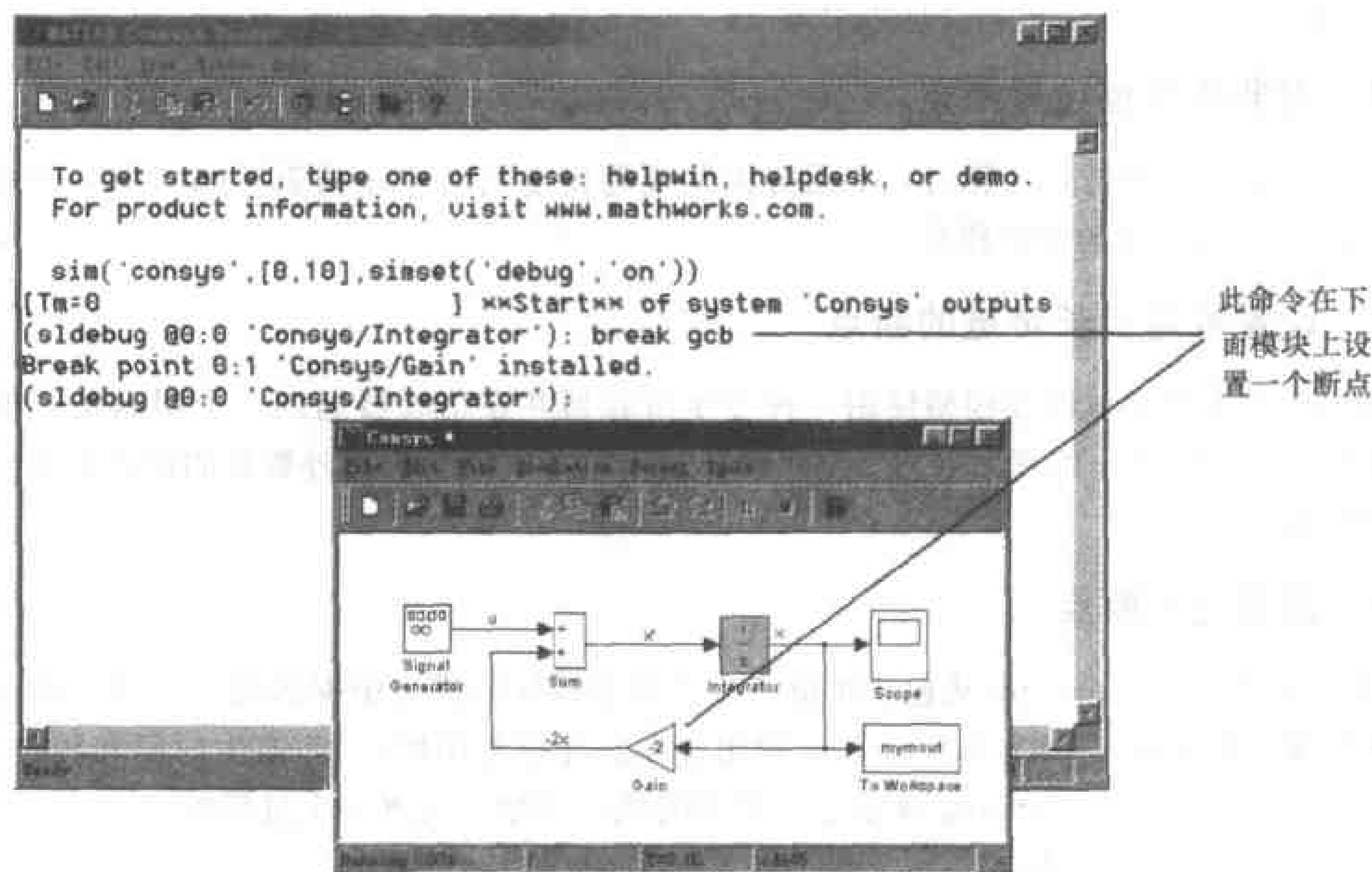


图 4.3　执行 break 命令后，在模块 Gain 设置断点

格式二：通过指数指定模块，输入：break s:b

其中，s:b 为模块的指数(参看 4.1.4“模块指数”)。

编者提示:系统不允许在虚拟模块上设置断点。所谓虚拟模块是这样的模块,它用纯图形表示模型的计算模块之间的组分或关系。若用户试图对一个虚拟模块设置断点,调试器就会发出警告。使用 slist 命令,用户可获得模型的非虚拟模块表(参见本章"显示模型的非虚拟模块")

2. 在模块的末尾处设置断点

bafter 命令可在一个非虚拟模块的末尾处设置一个断点。如同命令 break,用户可以通过图形或模块指数指定模块。

3. 清除模块上的断点

clearml 命令清除在一个模块起始处或末尾处的断点。可以通过输入模块指数或通过选定模型图中的模块并输入 clear 命令的自变量 gcb。

4.3.2 设置时间步断点

可以使用 tbreak 命令在某一个详尽的时间步上设置一个断点。tbreak 命令的惟一的自变量为时间量。该命令使调试器在指定时间随后的第一个时间步开始处中断仿真。例如,在调试模式下启动 Consys 并输入下列命令:

```
(sldebug @0:1 'Consys/Gain'): tbreak 9.5
(sldebug @0:1 'Consys/Gain'): continue
```

使调试器在时间步 9.599 99…开始处(如 continue 命令的输出所示)中断仿真,显示

```
[Tm = 9.599999999999998    ] * * Start * * of system 'Consys' outputs
(sldebug @0:0 'Consys/Integrator'):
```

4.3.3 对非限定值设置断点

nanbreak 命令在仿真计算出一个无穷大值或超出溢出时中止仿真。nanbreak 命令用于查明 Simulink 模型中的计算错误。

4.3.4 设置限定步长步进的断点

xbreak 命令使调试器在模型使用一种变步仿真器并在仿真器遇到一个限制仿真器可使用的步长的状态时中止仿真。本命令用于对出现需给仿真器提供额外数量的仿真时间步的模型进行调试。

4.3.5 设置过零断点

zcbreak 命令在 Simulink 从包含可能产生过零点模块的模型中探测到一个非采样过零点时使调试器中断仿真。中断后,Simulink 输出过零点在模型中的位置、产生时间和种类(上升沿或下降沿)。例如,在 zeroxing 演示程序模型的执行开始处设置一个过零断点:

```
>>sldebug zeroxing
[Tm = 0        ] * * Start * * of system 'zeroxing' outputs
(sldebug @0:0 'zeroxing/Sine Wave'): zcbreak
Break at zero crossing events is enabled.
```

并继续运行仿真

```
(sldebug @0:0 'zeroxing/Sine Wave'): continue
[Tm = 0.34350110879329        ] Breaking at block 0:5
[Tm = 0.34350110879329        ] Rising zero crossing on 3rd zcsignal in block 0:5
zeroxing/Saturation'
(sldebug @0:5 'zeroxing/Saturation'):
```

若模型不包含可能产生非采样过零的模块，则命令将输出告知信息。

4.4 显示仿真信息

Simulink 调试器提供一组命令用于让用户在运行模型期间显示模块状态、模块输入和输出，以及其信息。

4.4.1 显示模块 I/O

调试器提供三个命令用于显示模块 I/O。每一个命令显示一个指定模块的 I/O。它们之间的主要区别在于显示 I/O 的时间不同(见表 4.3)。

表 4.3 显示命令比较

命令	显示模块 I/O……
probe	立即
disp	在每一个断点处
trace	一经模块执行

1. probe 命令

probe 命令输出用户指定的当前模块的输入和输出(见表 4.4)。该命令在 MATLAB 命令窗口下显示模块的 I/O。

表 4.4 probe 命令及用途

命令	描　述
probe	进入或退出 probe 模式。在 probe 模式，调试器显示用户在模型图中选定的任何模块的当前输入和输出。键入任意命令会使调试器退出 probe 模式
probe gcb	显示选定模块的 I/O
probe s:b	输出系统编号和模块编号指定模块的 I/O(输入/输出)

probe 命令供用户需要检查还未显示 I/O 的模块的 I/O 时之用。例如，假设用户正在使用 step 命令逐一模块地运行模型。在每一次用户步进模型时，调试器显示当前模块的输入和输出。probe 命令也允许用户检查其他模块的 I/O。类似地，假设用户正在使用 next 命令以通过时间步步进模型。next 命令不显示模块 I/O。然而，若用户需要在输入 next 命令后检查

模块的 I/O，使用 probe 命令即可。

2. disp 命令

disp 命令使调试器在无论是否中断仿真的情况下显示一个指定模块的 I/O。用户既可以通过输入模块指数也可以通过在模型图中选定模块并输入作为 disp 命令自变量的 gcb 命令来指定该模块。用户使用 undisp 命令可以从调试器显示点清单中清除任意模块。例如，为清除模块 0:0，既可以在模型图中选定该模块并输入 undisp gcb，也可简单地输入 undisp 0:0。

disp 命令用于用户需要监控一个指定模块的 I/O 或在步进仿真程序时对模块组的监控。使用 disp 命令，用户可以指定用户需要监控的模块，调试器会在每一个时间步显示这些模块 I/O。

编者提示：当用户使用 step 命令运行逐一模块步进模型时，调试器总是显示当前模块 I/O。所以，如果用户仅仅对看一看当前模块 I/O 感兴趣，则不必使用 disp 命令。

3. trace 命令

trace 命令使调试器显示一个指定模块的 I/O，而不论 Simulink 是否给该模块赋值。这使得用户不必中止仿真即可获取一个模块 I/O 的全部记录。至于其他模块 I/O 的显示命令，用户可以通过输入该模块指数或在模型图中选定该模块的方法指定模块。用户可使用 untrace 命令从调试器的跟踪区中清除某个模块。

4.4.2 显示代数环信息

atrace 命令使调试器在解出模型的代数环(见 6.6.1 节中的“代数环”)的同时显示其相关信息(见表 4.5)。该命令接受一个指定信息显示量的单个变量。

表 4.5 atrace 命令及用途

命令	显示每一个代数环的……
atrace 0	无信息
atrace 1	环变量解，求解环的迭代次数，以及估计解误差
atrace 2	同级 1
atrace 3	级 2 加用于解回路的 Jacobian 矩阵
atrace 4	级 3 加回路变量的中解

4.4.3 显示系统状态

states 调试命令在 MATLAB 命令窗口列出系统状态的当前值。例如，下面的命令显示在第一个和第二个时间步后的 Simulink bounce demo 演示程序的结果。

```
>>sldebug bounce
[Tm = 0        ] * * Start * * of system 'bounce' outputs
(sldebug @0:0 'bounce/Position'): states
Continuous state vector (value,index,name):
10 0 (0:0 'bounce/Position')
15 1 (0:5 'bounce/Velocity')
```

```
(sldebug @0:0 'bounce/Position'): next
[Tm = 0.01        ] * * Start * * of system 'bounce' outputs
(sldebug @0:0 'bounce/Position'): states
Continuous state vector (value,index,name):
10.1495095 0 (0:0 'bounce/Position')
14.9019 1 (0:5 'bounce/Velocity')
```

4.4.4　显示集成信息

ishow 命令重复显示集成信息。激活该选项会使调试器每一次都输出关于调试器采取时间步或遇到一个限制某时间步长的状态的信息。对于第一种情况，调试器输出时间步长，例如：

```
[Tm = 9.996264188473381     ] Step of 0.01 was taken by integrator
```

对于第二种情况，调试器显示当前测定时间步长的状态，例如：

```
[Ts = 9.676264188473388     ] Integration limited by 1st state of
block 0:0 'bounce/Position'.
```

4.5　显示模型信息

除了提供有关仿真的信息外，调试器也向用户提供有关仿真模型的信息。

4.5.1　显示模型的模块执行顺序

在模型的初始化阶段，Simulink 即测定在仿真运行开始时执行模块的顺序。在仿真期间，Simulink 保存一个由执行顺序排定的模块序列，称之为排序表。用户可以通过在调试器命令提示符下键入 slist 随时显示排序表。slist 命令显示模型的模块执行顺序。排序表包括每个命令的模块指数：

```
---- Sorted list for 'Consys' [6 blocks, 6 nonvirtual blocks, directFeed = 0]
0:0 'Consys/Integrator' (Integrator)
0:1    'Consys/Gain' (Gain)
0:2    'Consys/Scope' (Scope)
0:3    'Consys/Signal Generator' (SignalGenerator)
0:4    'Consys/Sum' (Sum)
0:5    'Consys/To Workspace' (ToWorkspace)
(sldebug @0:0 'Consys/Integrator'):
```

4.5.2　显示一个模块

为了确定模型图中某个模块对应一个确定指数，在命令提示符下键入 bshow s:b，其中 s:b 为模块指数。bshow 命令打开包含模块的系统，并在系统窗口下选定模块。

4.5.3　显示模型的非虚拟系统

systems 命令输出正在调试的模型中的非虚拟系统表。例如，Simulink clutch demo 演示

程序包含了下面的系统：

```
>>sldebug clutch
[Tm = 0 ] * * Start * * of system 'clutch' outputs
(sldebug @0:0 'clutch/Clutch Pedal'): systems
0 'clutch'
1 'clutch/Locked'
2 'clutch/Unlocked'
```

编者提示：systems 命令不会列出实际上是纯图形的子系统，即模型图所呈现的子系统为 Subsystem 模块，但 Simulink 求出的子系统为一个父系系统的局部。在 Simulink 模型中，根系统和使能或触发子系统是真正的系统。而所有其他子系统是虚拟系统（即图形），因此它们不会出现在由 systems 命令生成的清单上。

4.5.4 显示模型的非虚拟模块

slist 命令显示模型中非虚拟模块的清单。清单按系统对模块分组。例如，下面一系列命令产生 Van der Pol (vdp) demo 模型中的非虚拟模块清单：

```
>>sldebug vdp
[Tm = 0        ] * * Start * * of system 'vdp' outputs
(sldebug @0:0 'vdp/Integrator1'): slist
- - - - Sorted list for 'vdp' [12 blocks, 9 nonvirtual blocks,
directFeed = 0]
0:0 'vdp/Integrator1' (Integrator)
0:1 'vdp/Out1' (Outport)
0:2 'vdp/Integrator2' (Integrator)
0:3 'vdp/Out2' (Outport)
0:4 'vdp/Fcn' (Fcn)
0:5 'vdp/Product' (Product)
0:6 'vdp/Mu' (Gain)
0:7 'vdp/Scope' (Scope)
0:8 'vdp/Sum' (Sum)
```

编者提示：slist 命令不会列出实际上是纯图形的子系统，即表示计算模块之间的关系或组分情况的模块。

4.5.5 显示含有潜在过零模块

zclist 命令输出一个在仿真期间可能出现非采样过零的模块清单。例如，zlist 输出下列离合器采样模型清单：

```
(sldebug @0:0 'clutch/Clutch Pedal'): zclist
2:3 'clutch/Unlocked/Sign' (Signum)
0:4 'clutch/Lockup Detection/Velocities Match' (HitCross)
0:10 'clutch/Lockup Detection/Required Friction
```

```
for Lockup/Abs' (Abs)
0:11 'clutch/Lockup Detection/Required Friction for
Lockup/ Relational Operator' (RelationalOperator)
0:18 'clutch/Break Apart Detection/Abs' (Abs)
0:20 'clutch/Break Apart Detection/Relational Operator'
(RelationalOperator)
0:24 'clutch/Unlocked' (SubSystem)
0:27 'clutch/Locked' (SubSystem)
```

4.5.6　显示代数环

ashow 命令加亮一个指定的代数环或包含指定模块的代数环。键入 ashow s#n 可加亮一个指定的代数环，其中 s 为包含回路的系统指数（详见本章"显示模型的执行顺序"），而 n 为该系统中的回路指数。键入 ashow gcb 可显示包含当前选定模块的回路。为显示包含一个指定模块的一个回路，可键入 ashow s:b，其中 s:b 为模块的指数。输入 ashow clear 可清除对模型图中的代数环加亮。

4.5.7　显示调试工具的设置

status 命令显示不同调试工具的选项，如条件断点。例如，下列命令显示对 vdp 模型初始调试设置：

```
>>sim('vdp',[0,10],simset('debug','on'))
[Tm = 0 ] * * Start * * of system 'vdp' outputs
(sldebug @0:0 'vdp/Integrator1'): status
Current simulation time: 0 (MajorTimeStep)
Last command: ""
Stop in minor times steps is disabled.
Break at zero crossing events is disabled.
Break when step size is limiting by a state is disabled.
Break on non-finite (NaN,Inf) values is disabled.
Display of integration information is disabled.
Algebraic loop tracing level is at 0.
```

4.6　调试器命令总汇

表 4.6 列出调试器命令。表中名为重复的列表示相应的命令是否提供按 Return 键重复命令的支持。关于各个命令的用途、命令参数及使用等信息，可参见随后各命令的详细说明。

表4.6 调试器命令总汇表

命令	简写	重复	描述
ashow	as	否	显示一个代数环
atrace	at	否	设置代数环的跟踪级
bafter	ba	否	在执行完一个模块后，插入一个断点
break	b	否	在执行一个模块之前，插入一个断点
bshow	bs	否	显示一个指定模块
clear	cl	否	清除模块断点
continue	c	是	继续仿真
disp	d	是	当仿真终止时，显示一个模块的 I/O
help	? 或 h	否	显示调试器命令的帮助
ishow	i	否	使能或禁止显示集成信息
minor	m	否	使能或禁止小时间步模式
nanbreak	na	否	设置或清除非限定值中断
next	n	是	转到下一个时间步的开始
probe	p	否	显示一个模块 I/O
quit	q	否	中止仿真
run	r	否	运行仿真至结束
slist	sli	否	列出一个模型的非虚拟模块清单
states	state	否	显示当前状态值
status	stat	否	显示实际调试选项
step	s	是	步进到下一个模块
stop	sto	否	停止仿真
systems	sys	否	列出一个模型的非虚拟系统清单
tbreak	tb	否	设置或清除一个时间断点
trace	tr	是	显示执行模块的 I/O
undisp	und	是	从调试器显示点清单取消一个模块
untrace	unt	是	从调试器跟踪区取消一个模块
xbreak	x	否	当调试器遇到一个限制步长状态时中断
zcbreak	zcb	否	在非采样过零时间发生时中断
zclist	zcl	否	列出包含非采样过零的模块清单

下面给出各个命令的详细描述。

4.6.1　ashow 命令

(1) 命令用途：显示代数环。

(2) 命令格式：ashow <gcb | s:b | s#n | clear>

(3) 变量取值：

① s:b 模块的系统指数 s，模块指数 b；

② gcb 当前模块；

③ s#n 系统 s 中编号 n 的代数环；

④ clear 清除回路色彩的开关。

(4) 命令描述：ashow gcb 或 ashow s:b 加亮包含指定模块的代数环。ashow s#n 加亮系统 s 中第 n 个代数环。ashow clear 从模型图中清除代数环的加亮。

(5) 相关命令：atrace，slist。

4.6.2　atrace 命令

(1) 命令用途：设置代数环跟踪等级。

(2) 命令格式：atrace level

(3) 变量取值及相应功能：

level：跟踪等级(0＝不跟踪；4＝所有)

① atrace 0 无信息；

② atrace 1 回路变量解、解出回路的迭代次数以及估计解误差；

③ atrace 2 同等级 1；

④ atrace 3 等级 2＋用于解回路的 Jacobian 矩阵；

⑤ atrace 4 等级 3＋环变量的中解。

(4) 命令描述：设置仿真代数环的跟踪等级。

(5) 相关命令：systems，states。

4.6.3　bafter 命令

(1) 命令用途：在一个模块执行后插入一个断点。

(2) 命令格式：bafter gcb

bafter s:b

(3) 变量取值：

① s:b 模块的系统指数 s，模块指数 b；

② gcb 当前模块。

(4) 命令描述：bafter 命令在指定模块执行后插入一个断点。

(5) 相关命令：break，xbreak，tbreak，nanbreak，zcbreak，slist。

4.6.4　break 命令

(1) 命令用途：在一个模块执行前插入一个断点。

(2) 命令格式：break gcb

break s:b

(3) 变量取值:

① s:b 模块的系统指数 s,模块指数 b;

② gcb 当前模块。

(4) 命令描述:break 命令在指定模块执行前插入一个断点。

(5) 相关命令:bafter, tbreak, xbreak, nanbreak, zcbreak, slist 命令。

4.6.5 bshow 命令

(1) 命令用途:显示一个指定模块。

(2) 命令格式:bshow s:b

(3) 变量取值:s:b:模块的系统指数 s,模块指数 b。

(4) 命令描述:本命令打开包含指定模块的模型窗口,并选中该模块。

(5) 相关命令:slist 命令。

4.6.6 clear 命令

(1) 命令用途:清除一个模块上的断点。

(2) 命令格式:clear gcb

clear s:b

(3) 变量取值:

① s:b 模块的系统指数 s,模块指数 b;

② gcb 当前模块。

(4) 命令描述:clear 命令清除一个指定模块上的断点。

(5) 相关命令:bafter, slist 命令。

4.6.7 continue 命令

(1) 命令用途:继续仿真。

(2) 命令格式:continue

(3) 命令描述:continue 命令从当前断点继续运行仿真,直到遇到下一个断点或最终时间步。

(4) 相关命令:run, stop, quit 命令。

4.6.8 disp 命令

(1) 命令用途:仿真停止时,显示一个模块的 I/O。

(2) 命令格式:disp gcb

disp s:b

disp

(3) 变量取值:

① s:b 模块的系统指数 s,模块指数 b;

② gcb 当前模块。

(4) 命令描述:disp 命令将一个模块登记为一个显示点。只要仿真保持中止状态,调试器就会在 MATLAB 命令窗口上显示所有显示点的输入和输出。调用无自变量的 disp 命令显示显示点的清单。使用 undisp 命令清除对一个模块的登记。

(5) 相关命令：undisp，slist，probe，trace 命令。

4.6.9　help 命令

(1) 命令用途：显示调试器命令的帮助。

(2) 命令格式：help

(3) 命令描述：help 命令在命令窗口上显示调试器命令清单。清单包括命令格式以及每一个命令的简要描述。

4.6.10　ishow 命令

(1) 命令用途：使能或禁止显示集成信息。

(2) 命令格式：ishow

(3) 命令描述：ishow 命令在仿真期间，转换显示集成信息。

(4) 相关命令：atrace 命令。

4.6.11　minor 命令

(1) 命令用途：使能或禁止小时间步模式。

(2) 命令格式：minor

(3) 命令描述：minor 命令使调试器进入或退出小时间步模式。在小时间步模式下，step 命令在小时间步内进行模块步进仿真。在小时间步模式下，在执行完模型的模块排序表中的最后一个模块后，若所有小时间步均为当前大时间步的子步，则 step 命令使仿真进行到下一个小时间步；否则 step 命令将使仿真进到下一个大时间步中的第一个小时间步上。

(4) 相关命令：step 命令。

4.6.12　nanbreak 命令

(1) 命令用途：设置或清除非限定值中断模式。

(2) 命令格式：nanbreak

(3) 命令描述：nanbreak 命令使调试器只要当仿真遇到一个非限定(NaN 或 Inf)值时中断。若设置了非限定中断模式，则 nanbreak 命令就将该模式清除。

(4) 相关命令：break，bafter，xbreak，tbreak，zcbreak 命令。

4.6.13　next 命令

(1) 命令用途：转到下一个时间步的开始。

(2) 命令格式：next

(3) 命令描述：next 命令计算当前时间步中未计算的模块，停在下一个时间步的开始。在执行 next 命令后，调试器使在下一个时间步内计算的模块加亮显示，并显示下一个时间步的时间。

(4) 相关命令：step 命令。

4.6.14　probe 命令

(1) 命令用途：显示一个模块的 I/O。

(2) 命令格式：probe [<s:b | gcb>]

(3) 变量取值：

① s:b 模块的系统指数 s，模块指数 b；

② gcb 当前模块。

(4) 命令描述：probe 命令使调试器进入或退出探针模式。在探针模式下，调试器显示任意用户选定模块的 I/O。键入任意命令，可退出探针模式。probe gcb 显示当前选定模块的 I/O；probe s:b 显示指数是 s:b 的模块 I/O。

(5) 相关命令：disp，trace 命令。

4.6.15 quit 命令

(1) 命令用途：放弃仿真。

(2) 命令格式：quit

(3) 命令描述：quit 命令中止当前仿真。

(4) 相关命令：stop 命令。

4.6.16 run 命令

(1) 命令用途：运行后续仿真。

(2) 命令格式：run

(3) 命令描述：run 命令从当前断点开始运行仿真直至最后一个时间步。该命令忽视所有断点和显示点。

(4) 相关命令：continue，stop，quit 命令。

4.6.17 slist 命令

(1) 命令用途：列出一个模型的非虚拟模块清单。

(2) 命令格式：slist

(3) 命令描述：slist 命令列出正在调试的模型中的非虚拟模块清单。清单出示每一个列出模块的模块指数和名。

(4) 相关命令：systems 命令。

4.6.18 states 命令

(1) 命令用途：显示当前状态值。

(2) 命令格式：states

(3) 命令描述：states 命令列出模型的当前状态清单。显示每一个状态的值、指数和名。

(4) 相关命令：ishow 命令。

4.6.19 systems 命令

(1) 命令用途：列出一个模型的非虚拟系统。

(2) 命令格式：systems

(3) 命令描述：systems 命令在 MATLAB 命令窗口列出一个模型的非虚拟系统清单。

(4) 相关命令：slist 命令。

4.6.20　status 命令

(1) 命令用途：显示实际调试工具选项。

(2) 命令格式：status

(3) 命令描述：status 命令显示实际使用的调试工具选项清单。

4.6.21　step 命令

(1) 命令用途：步进到下一个模块。

(2) 命令格式：step

(3) 命令描述：step 命令计算下一个在当前时间步未计算的模块。执行完 step 命令后，调试器将下一个要计算的模块和输出信号连线加亮。同时，在调试器命令行提示中显示下一个模块名。

(4) 相关命令：next 命令。

4.6.22　stop 命令

(1) 命令用途：中止仿真。

(2) 命令格式：stop

(3) 命令描述：stop 命令中止仿真。

(4) 相关命令：continue，run，quit 命令。

4.6.23　tbreak 命令

(1) 命令用途：设置或清除一个时间断点。

(2) 命令格式：tbreak t

　　　　　　tbreak

(3) 命令描述：tbreak 命令在指定的时间步上设置一个断点。若在指定时间上已存在一个断点，则 tbreak 命令将清除该断点。若命令未指定时间值，tbreak 则在当前时间步上重复设置一个断点。

(4) 相关命令：break，bafter，xbreak，nanbreak，zcbreak 命令。

4.6.24　trace 命令

(1) 命令用途：显示执行模块的 I/O。

(2) 命令格式：trace gcb

　　　　　　trace s:b

(3) 变量取值：

① s:b 模块的系统指数 s，模块指数 b；

② gcb 当前模块。

(4) 命令描述：trace 命令将一个模块登记为一个跟踪点。在每执行一个登记模块时，调试器显示该模块 I/O。

(5) 相关命令：disp，probe，untrace，slist 命令。

4.6.25　undisp 命令

(1) 命令用途：从调试器的显示点清单上清除一个模块。

(2) 命令格式:undisp gcb

undisp s:b

(3) 变量取值:

① s:b 模块的系统指数 s,模块指数 b;

② gcb 当前模块。

(4) 命令描述:undisp 命令从调试器的显示点清单上清除指定模块。

(5) 相关命令:disp, slist 命令。

4.6.26 untrace 命令

(1) 命令用途:从调试器的跟踪点清单上清除一个模块。

(2) 命令格式:untrace gcb

untrace s:b

(3) 变量取值:

① s:b 模块的系统指数 s,模块指数 b;

② gcb 当前模块。

(4) 命令描述:untrace 命令从调试器的跟踪点清单上清除指定模块。

(5) 相关命令:trace, slist 命令。

4.6.27 xbreak 命令

(1) 命令用途:当调试器遇到一个限制步长状态时中断。

(2) 命令格式:xbreak

(3) 命令描述:xbreak 命令在调试器遇到一个限定仿真器采用的时间步的步长时暂停模型的执行。若此时为 xbreak 模式,执行 xbreak 命令将清除该模式。

(4) 相关命令:break, bafter, zcbreak, tbreak, nanbreak 命令。

4.6.28 zcbreak 命令

(1) 命令用途:重复在非采样过零事件发生时的中断。

(2) 命令格式:zcbreak

(3) 命令描述:zcbreak 命令使调试器在一个非采样过零事件发生时产生中断。若此时为过零中断模式,则执行 zcbreak 命令将取消该模式。

(4) 相关命令:break, bafter, xbreak, tbreak, nanbreak, zclist 命令。

4.6.29 zclist 命令

(1) 命令用途:列出包含非采样过零模块的清单。

(2) 命令格式:zclist

(3) 命令描述:zclist 命令输出可能发生非采样过零的模块清单。清单在 MATLAB 命令窗口上输出。

(4) 相关命令:zcbreak 命令。

第5章 Simulink 仿真运行与结果分析

模型建立以后的任务是仿真和结果分析，由此检验模型的正确性和实用性。但在仿真的过程中，有很多因素，如用于仿真的仿真器、步长、时间等均会影响仿真的结果。本章的内容将有助于解决这些问题。

主要内容：Simulink 仿真模型的优化方法和结果分析方法。

学习目的：通过本章的学习，掌握对经过动态调试纠错的仿真模型进行优化仿真与结果分析，最终得到自己满意的仿真结果。

5.1 仿真的运行方式比较

运行仿真一般分为两种方式，即通过 Simulink 的菜单命令运行仿真和在 MATLAB 命令窗口中输入命令运行仿真。当然，在实际应用中，许多用户在开发和优化模型时，使用菜单命令，然后在 MATLAB 命令窗口输入命令来分批地运行仿真。读者可以根据自己喜好，进行选择。

5.1.1 使用菜单命令

因为菜单命令是交互式使用的，所以运行仿真很容易。使用菜单命令的主要优点在于使用者不需记住命令句法，只要选择一个常微分方程(ODE) 仿真器(solver)，并定义仿真参数即可。另外一个显著的特点是在一个仿真运行期间，能执行一些交互式操作。如：

(1) 利用对话框可以改变许多仿真参数，包括停止时间、仿真器，以及最大步长等；

(2) 可方便地选择不同类型的仿真器进行仿真计算和比较；

(3) 可同时运行多个仿真；

(4) 在一条线上单击，用一个浮动的(未连接的)Scope 或 Display 模块查看所载信号；

(5) 可以改变模块的参数，只要下列参数不会发生变化：

① 状态、输入或输出的数目；

② 采样时间；

③ 过零检测数；

④ 任何模块参数的矢量长度；

⑤ 内部模块工作矢量长度。

编者提示：在仿真的运行过程中，不能改变模型结构，如添加或删除模型中的模块或线。如果需要进行这类更改，必须先停止仿真，更改后，再重新运行仿真，查看结果的变化。

5.1.2　从命令行运行仿真

从 MATLAB 命令窗口运行仿真比使用菜单命令运行仿真有以下优点：

(1) 可以与 Simulink 模型一样，对 M 文件模型和 Mex 文件模型进行仿真。

(2) 在 M 文件中嵌入运行仿真程序，系统支持仿真和模块参数的交互式更改。

下面分别对上述两种运行方式做详细介绍。

5.2　使用菜单命令运行仿真

本节讨论如何使用 Simulink 菜单命令和仿真参数(Simulation Parameters)对话框来运行仿真。

5.2.1　设置仿真参数和选择仿真器(Solver)

设置仿真参数和选择仿真器需通过从 Simulation 菜单中选择 Parameters 项。Simulink 会显示管理仿真参数的 Simulation Parameters(仿真参数)对话框：

(1) Solver 选项用于设置仿真的开始和停止时间，选择仿真器(solver)的类型和设置 solver 参数，选择输出项。

(2) Workspace I/O 选项用于管理 MATLAB 工作空间的输入和输出。

(3) Diagnostics(诊断)选项用于选择设置仿真过程中警告信息的类型。

关于对话框的选项，包括设置参数，将在"仿真参数对话框"中详细讨论。

可以将指定参数设置为有效的 MATLAB 表达式，包括常数、工作空间变量名、MATLAB 函数，以及数学运算符。

5.2.2　应用仿真参数

在设置仿真参数并选择了仿真器之后，就可以将它们加到自己的模型中了。点击 Apply 按钮可加入参数。按 Close 按钮加入参数并关闭对话框。

5.2.3　运行仿真

在将仿真参数和仿真器加入到自己的模型之后，就可以开始运行仿真。从 Simulation 菜单中选择 Start 项，仿真开始运行。还可以使用快捷键 Ctrl+T。在选择 Start 后，该菜单项变为 Stop。

编者提示：必须确保在开始仿真前，仿真模型窗口为当前激活状态。

终止仿真：从 Simulation 菜单中选择 Stop 项，或者与开始运行仿真一样，使用键盘快捷键 Ctrl-T 来终止仿真。

挂起(暂停)仿真：可以从 Simulation 菜单中选择 Pause。在选择 Pause 后，该菜单项变为 Continue。用户可以选择 Continue，继续进行被临时中断的仿真。

计算机以蜂鸣声来表示仿真完成。

如果模型包含有向一个文件或工作空间写数据的模块，或者如果用户在仿真参数对话框中选择输出项，则当仿真终止或挂起时，Simulink 就会对当前数据进行写操作。

5.2.4　仿真诊断(Simulation Diagnostics)对话框

在仿真运行过程中，如果发生错误，Simulink 就会停止仿真并在仿真诊断对话框中显示错误信息。

仿真诊断对话框如图 5.1 所示。该对话框有两个子窗。上面的子窗显示每一个错误的信息，共有 5 列：

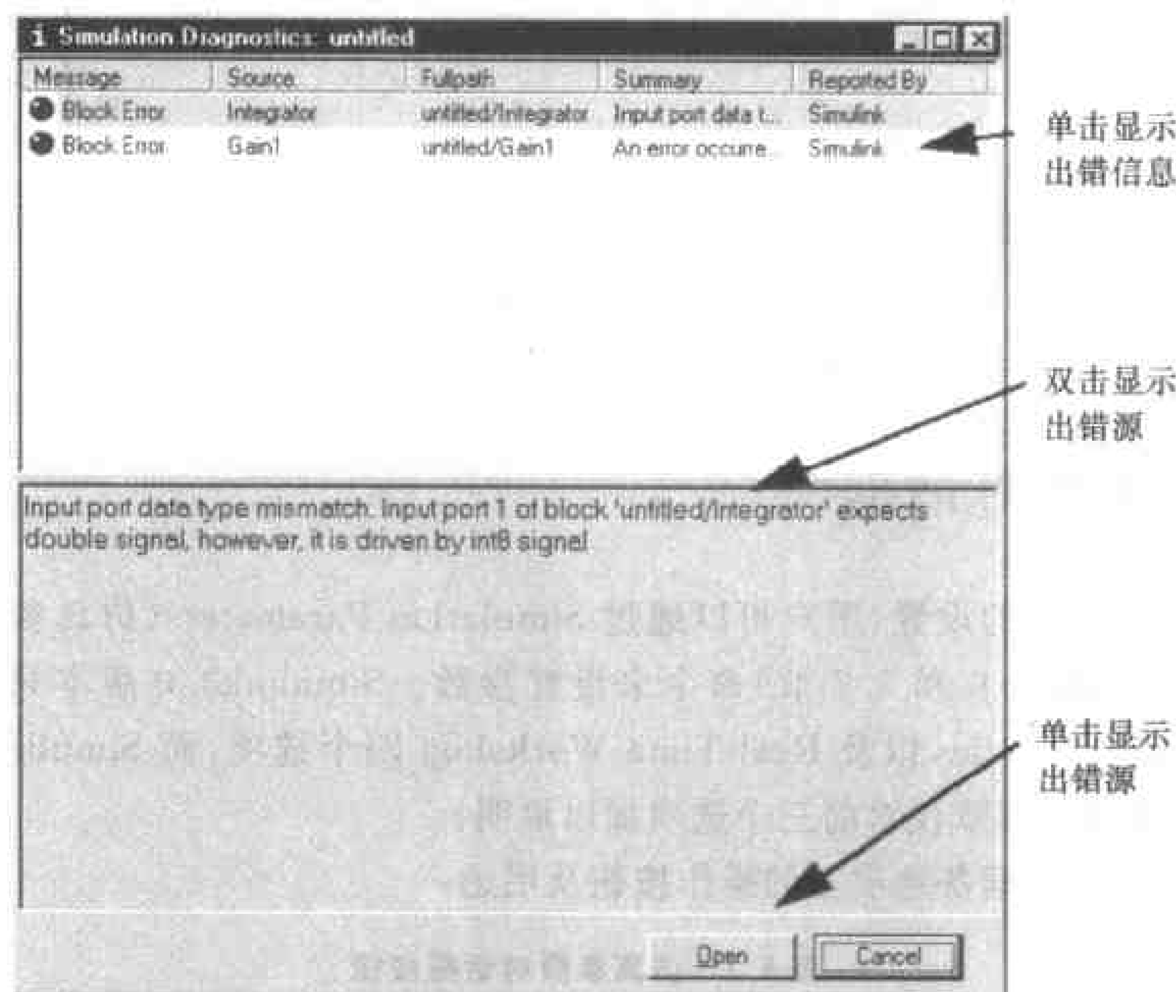


图 5.1　仿真诊断对话框上的信息

① Message：信息类型(如：模块错误、警告、记录)；

② Source：引起该错误的模型元素名(如：模块名)；

③ Fullpath：引起该错误的元素的路径；

④ Summary：出错信息摘要；

⑤ Reported by：报告该错误的出处(如：Simulink，Stateflow，Real-Time Workshop 等)。

下面的子窗在起始时只列出对应第一个错误的所有信息。单击上子窗格中的其他某一个错误条目，则立即显示其对应的信息内容。

除了显示仿真诊断对话框之外，如果有必要，Simulink 还会打开包含该错误源的模型图，并高亮显示，如图 5.2 所示。

类似地，还可通过双击上子窗中的相应出错信息条，或在下子窗，双击出错信息中的出错源名(高亮蓝色)，或点击该对话框中的 Open 按钮，来显示其他错误源，见图 5.1 中文标识。

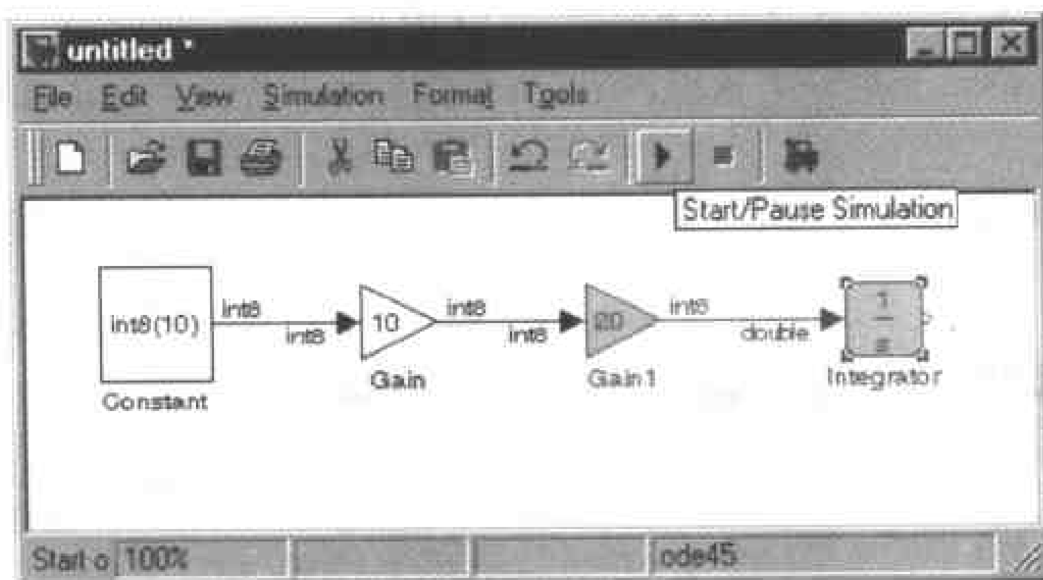


图 5.2　在模型图中高亮显示错误源

5.3　仿真参数对话框

本节介绍仿真参数的设置，用户可以通过 Simulation Parameters（仿真参数）对话框或使用 sim 和 simset（本章 5.6 和 5.7 节）命令来设置参数。Simulink3.0 版本只提供了 Solver，Workspace I/O，Diagnostics 以及 Real-Time Workshop 四个选项，而 Simulink4.0 版本又添加了 Advanced 选项。本章仅对前三个选项加以说明。

表 5.1 给出了对话框各选项上的操作按钮及用途：

表 5.1　仿真参数对话框按钮

按钮	操　　作
Ok	应用参数值并关闭对话框。在仿真过程中，参数值立即应用
Cancel	恢复参数值到打开对话框时的状态，并关闭对话框
Help	显示对话框页的帮助文档
Apply	应用当前参数值并保持对话框为打开状态。在仿真过程中，参数值立即应用

5.3.1　Solver 选项及其设置

Solver 选项在第一次从 Simulation 菜单中选择 Parameters 项，或选择 Solver 标签时出现，如图 5.3 所示。通过该选项可设置仿真开始和停止时间、选择仿真器并指定参数（请特别关注）和设置输出选项等。下面具体说明。

1. 仿真时间（Simulation time）

包括起始时间（Start time）和停止时间（Stop time），它们的差即代表仿真时间。

设置起始时间和停止时间，在 Start time 和 Stop time 编辑框内输入相应的值即可。默认起始时间是 0.0 s，默认停止时间是 10.0 s。

值得注意的是，设置的仿真时间与实际运行的时间并不一致。例如，设置一个仿真时间为 10 s，通常的运行仿真时间并没有 10 s。仿真运行时间与许多因素有关，包括模型的复杂性，

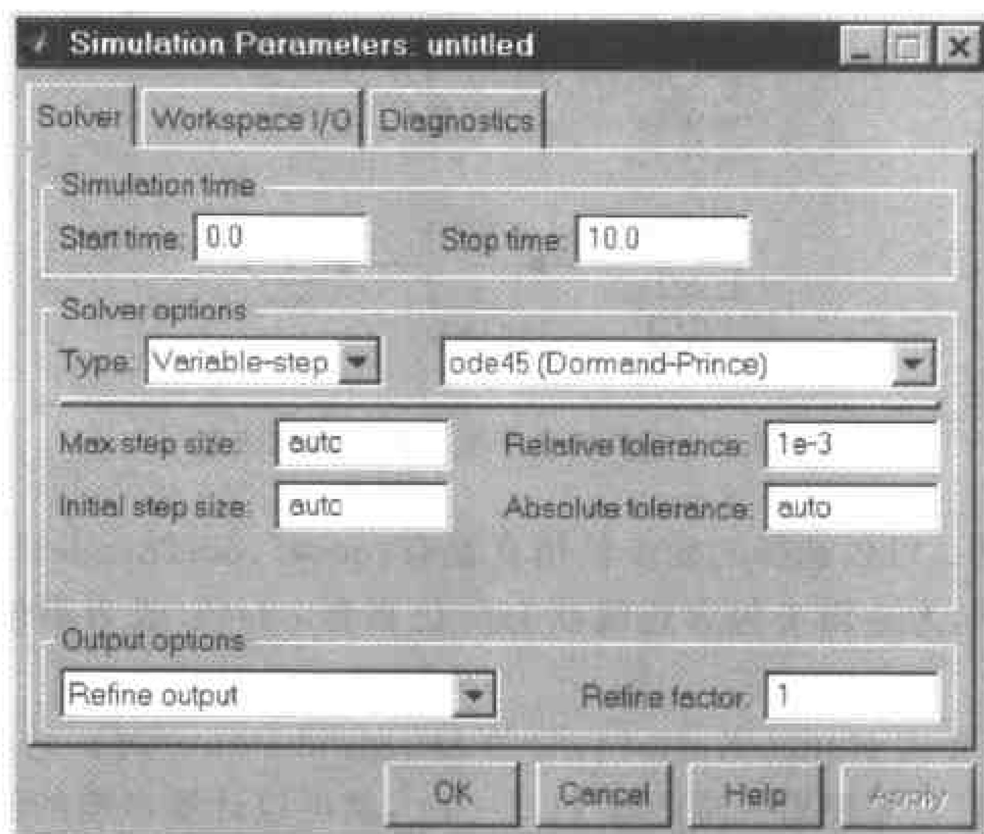


图5.3 Solver选项的参数设置对话框

仿真器的步长大小,以及计算机时钟速度等。

2. 仿真器选项(Solver options)

Simulink模型的仿真包括常微分方程(ODEs)数值积分。Simulink提供了许多求解这类方程式的仿真器。由于动态系统特性的多样性,针对一些特别的问题,某些仿真器的算法会比其他解法更有效。因此,为了获得更高的精确度并快速得出结果,应该特别关注仿真器的选择和参数设置。

Type项用于选择变步长(variable-step)或固定步长(fixed-step)仿真器,左栏为步长类型,右栏用于选择仿真器的算法。变步长仿真器在仿真期间可以调整步长,并提供误差控制和过零检测。固定步长仿真器在仿真过程中,其步长是不变的,没有误差控制和过零检测。

(1) 系统内置仿真器。如果没有选择一种仿真器,Simulink基于下列的条件,选择一个默认的仿真器:

• 如果当前模型包含连续状态(系统),则使用ode45。ode45是一种通用型仿真器。然而,如果知道自己的系统是刚性的,并且在使用ode45后不能给出可接受的结果,应尝试ode15s。关于刚性的定义,参看后面的"变步长仿真器"。

• 如果模型是离散的,Simulink使用叫做变步长discrete(离散)仿真器,并且显示没有使用ode45的信息提示。同时,Simulink又提供一个叫做固定步长discrete仿真器。图5.4的模型显示了两种discrete解法的不同。

由于采样时间分别为0.5 s和0.75 s,则该模型的基本采样时间是0.25 s。变步长仿真器与固定步长仿真器的不同之处在于它们各自产生的时间矢量的长度是不同的:

固定步长的discrete仿真器产生的时间矢量为:[0.0 0.25 0.5 0.75 1.0 1.25 ...]

变步长discrete仿真器产生的时间矢量为:[0.0 0.5 0.75 1.0 1.5 2.0 2.25 ...]

固定步长的discrete仿真器的步长是基本采样时间,而变步长discrete仿真器采用了最大可能的步长。

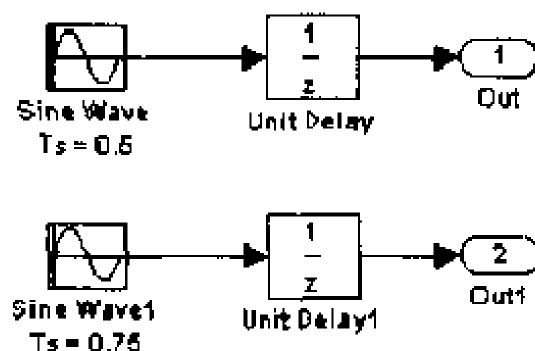


图5.4 变步长会产生不同长度的矢量

① 变步长仿真器。可供选择的变步长仿真器有:ode45,ode23,ode113,ode15s,ode23s和discrete。默认情况下,为适用于包含连续状态的仿真器ode45,或为不包含连续状态的discrete仿真器:

- ode45是基于显式Runge-Kutta(4,5)的Dormand-Prince对。它是一个单步仿真器,即:在计算 $y(t_n)$ 时,它只需要前一步的解 $y(t_n-1)$,就可以计算当前的值。一般而言,ode45是解决许多问题的“最佳”的首选仿真器。
- ode23是基于Bogacki和Shampine的显函数Runge-Kutta(2,3)对。在宽容差和存在轻度刚性时,可能比ode45更有效。ode23也是一个单步仿真器。
- ode113是变阶Adams-Bashforth-Moulton PECE仿真器,在处理严格容差问题时,会比ode45更有效。ode113是一个多步仿真器,也就是说,通常需要计算前几步的结果来计算当前值。
- ode15s是基于数值微分公式(NDFs)的变阶仿真器,数值微分公式与倒微分公式BDFs(也称作Gear法)相关,但比其更有效。与ode113类似,ode15s也是一个多步仿真器。如果怀疑一个问题可能是刚性的,或者使用ode45的效率极差或失败时,可以考虑尝试ode15s。
- ode23s是基于2阶修正Rosenbrock公式,由于它是单步仿真器,在处理有宽容差的问题时,可能比ode15s更有效。它能解决一些使用ode15s无效的刚性问题。
- ode23t采用自由内插法实现的梯形算法(trapezoidal rule)。适用于解决仅有适度刚性,且需要无数值衰减的问题。
- ode23tb是TR-BDF2方法的实现,即,隐函数Runge-Kutta公式,其一级采用梯形法,二级采用二阶倒向微分公式。通过构造,两级的计算都用到了相同的迭代矩阵。与ode23s一样,该仿真器在处理有宽容差的问题时,会比ode15s更有效。
- discrete(变步)是在Simulink检测到模型中没有连续状态时选择的仿真器。

编者提示:对于刚性问题,其解法会在相对于积分间隔极短的时间刻度上变化,但是感兴趣的解法却在一个较长的时间刻度上变化。对于不是专门为解决刚性问题而设计的或相对于时间间隔解法变化缓慢的方法,其效率就不高,因为它们的时间步长太短,不能适应解决可能最快的变化。Jacobian矩阵用于产生数字ode15s和ode23s。欲知更多的信息,请参看Shampine, L. F.著的“Numerical Solution of Ordinary Differential Equations”, Chapman & Hall, 1994出版。

② 固定步长仿真器。可供选择的固定步长算法有:ode5, ode4, ode3, ode2, ode1和discrete:

- ode5:固定步长的ode45,即Dormand-Prince公式。

- ode4：RK4，即四阶 Runge-Kutta 公式。
- ode3：固定步长的 ode23，Bogacki-Shampine 公式。
- ode2：Heun 法，也称为改进的 Euler 公式。
- ode1：Euler 法。
- discrete (fixed-step)：一种不含积分运算的固定步长仿真器。适合于没有连续状态，不需过零检测和误差控制的模型。

如果认为仿真结果并不令人满意，请阅读下面的内容。

(2) 选择仿真器中应注意的问题。对于大多数模型，默认的 Solver 参数一般都可以提供精确并且有效的结果。然而对一些问题，调整该参数能提高仿真的性能(有关调整这些参数的更多信息，参看“提高仿真性能和精确度”一节)。用户可以通过改变 Solver 面板上的参数值来调整所选的解法。

① 步长。对于变步长算法，用户可以设置最大步长值和建议初始步长参数。Simulink 4.0版本增加了最小步长值(Min step size)。这些参数的默认值是自动决定的，指示值为 auto。

对于固定步长的算法，用户可以设置固定步长大小，其默认值也是 auto。

- 最大步长值(Max step size)。该参数控制当前仿真器可以采用的最大时间步。默认值由开始和停止时间决定：

$$h_{max} = \frac{t_{stop} - t_{start}}{50}$$

通常，默认最大步长值是足够的。如果担心仿真器在仿真计算中会丢失重要的过程，可以改变此参数以避免采用过大的步长。如果仿真时间很长，默认步长可能会太大，以致求不出解。另一方面，如果模型包含周期的或近似周期的过程，把最大步长设为该周期的分数值为好(比如 1/4)。一般情况下，如果需要更多的输出(值)点，改变改善因子(refine factor)，而不需改变最大步长值。更多的信息，请看下面的“改善输出”。

- 初始步长值(Initial step size)。默认时，仿真器在起始时间通过检查状态微分，选择初始步长。如果第一个步长太大，则仿真器会越过这个重要的动作。初始步长参数是一个建议步长。仿真器先尝试此步长，但是如果不能满足误差标准，会减小步长。

② 误差容限。仿真器使用标准局部误差控制技术来监视每步的误差。在每步期间，仿真器在每步的结尾计算状态值，并测定局部误差(local error)，即状态值的估计误差。然后将局部误差与可接受误差(acceptable error)比较，其中可接受误差是相对误差容限(rtol)和绝对误差容限(atol)的函数。如果对于任何状态，该误差都大于可接受误差，则仿真器就会减小步长，并重新计算。

- 相对误差容限规定了每个状态相对于步长的误差大小，用百分比表示一个状态值。默认值为 1e-3，即计算状态要精确到 0.1%以内。
- 绝对误差容限是一个阈误差值，当测量的状态值接近零时，代表可接受误差。

第 i 个状态的误差 e_i，必须满足

$$e_i \leqslant \max(\mathrm{rtol}\,x \mid x_i \mid, \mathrm{atol}_i)$$

图 5.5 显示了状态曲线，和由相对误差容限或绝对误差容限决定可接受误差的区域。

如果指定 auto(默认)，Simulink 将每个状态的初始绝对误差容限设置为 1e-6。随着仿真

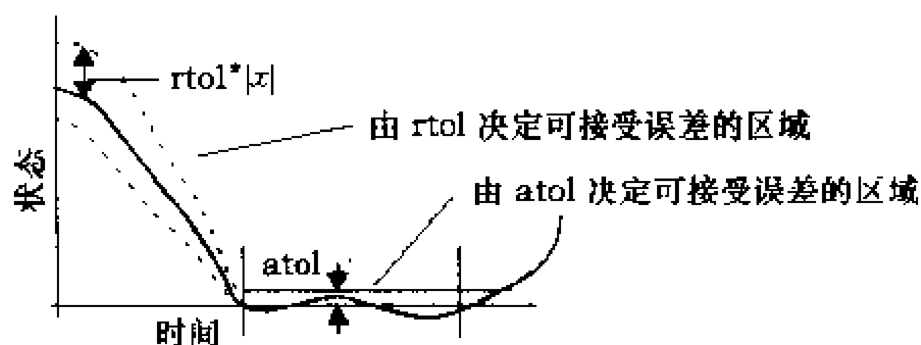


图 5.5　状态曲线

的进行，Simulink 将每个状态的绝对误差容限置位到该状态曾设定的最大值，该值是该状态的相对误差容限的许多倍。因此，如果一个状态从 0 到 1 且 reltol 是 1e-3，则在仿真的末尾，abstol 也被设置为 1e-3。如果一个状态从 0 到 1 000，那么 absto 被设置为 1。

如果计算设置不适当，用户可以自己确定一适当的设置。但可能需多次运行仿真来确定适当的绝对误差容限值。如果该状态的大小变化幅度很大，最好为不同的状态指定不同的绝对误差容限值。这可以在 Integrator 模块对话框上完成设置。

③ ode15s 的最大阶(Maximum Order)

ode15s 仿真器是基于一到五阶数值微分公式的。虽然高阶公式精确度高，但稳定性比较差。如果用户模型是刚性的且需要更好的稳定性，应将最大阶减少到 2(数值微分公式 A 级稳定的最高阶)。当选择 ode15s 仿真器时，对话框将显示这个参数。

也可以尝试使用 ode23s 仿真器，它是一种固定步长、低阶(A 级稳定)的仿真器。

④ 任务选项

如果选择固定步长的仿真器，Simulation Parameters 对话框的 Solver 页显示一个下拉 Mode 选项列表。该列表提供下列仿真模式可供选择。

• MultiTasking。这个模式在它检测到模块之间有非法的采样率转换，即直接连接的模块之间有着不同的采样率时，显示错误提示。在实时多任务系统中，任务之间非法的采样率转换可能导致某一个任务的输出无法为另一个任务使用。通过对这种转换的检验，多任务模式将有助于创建有效的真实多任务系统模型，其中用户模型部分代表并发任务。

使用采样率转换模块来消除非法的采样率速率转换。Simulink 提供了两个这样的模块：Unit Delay(单位延迟模块)和 Zero-Order Hold(零阶保持模块)模块。要消除非法的慢-快采样率转换，插入一个 Zero-Order Hold 模块，使其在慢输出端口和快输入端口之间以慢速率运行。要消除非法的快—慢采样率转换，需插入一个 Zero-Order Hold 模块，使其在快输出端口和慢输入端口之间以慢速率运行。详细信息，参看 Real-Time Workshop 关于多采样率模型的说明。

• SingleTasking。该模式不检查模块之间的采样率转换。当建立单任务系统模型时，本模式是有用的。在这样的系统中，任务同步不是一个问题。

• Auto。这一项是当所有模块以同一个速率运行时，Simulink 采用 SingleTasking(单任务)模式，而当模型包含不同速率模块时，采用 MultiTasking(多任务)模式。

3. 设置输出选项(Output options)

对话框的 Output Options 区域可以控制仿真产生的输出。有三个弹出式选项可供选择：

“改善输出”,“产生附加输出”以及“只产生指定输出”。

• 改善输出(Refine output)。当仿真输出太粗糙时,该选择项提供附加的输出点。这个参数在时间步长之间提供一个整数的输出点,例如,改善因子为 2 时,不但在时间步长点提供输出,在时间步长之间的中间点也提供输出。默认的改善因子为 1。

为了得到较为平滑的输出,改变改善因子比减少步长要快得多。当改变改善因子时,仿真器通过计算这些点的连续延拓公式,产生相应的附加点。改变改善因子并不改变步长。

改善因子很适用于变步长仿真器,尤其是当使用 ode45 时。ode45 仿真器需要大步长,当绘制仿真输出时,可能会发现当前使用的仿真器的输出并不是很平滑。在这种情况下,使用大的改善因子再次运行该仿真,如改善因子值为 4 时,输出结果会相当平滑。

• 产生附加输出(Produce additional output)。该选项可以直接指定输出的附加时刻。当选中该项时,Simulink 在 Solver 选项内显示 Output Times 编辑框,可以输入用于计算附加时刻或附加时刻矢量的 MATLAB 表达式。附加输出是在附加时刻使用连续延拓公式产生的。不同于改善因子的是,这种方式是改变仿真步长以使时间步长与指定的附加输出的时刻一致。

• 只产生指定输出(Produce specified output only)。该项只在指定时刻产生仿真输出。这种方式改变仿真步长以使时间步长与指定输出时刻一致。在比较不同的仿真时,要确保在同一个时刻产生输出时,该项是很有用的。

• 三种输出选项的比较。

我们看一个例子,假设仿真产生输出的时刻为:

0, 2.5, 5, 8.5, 10

选择 Refine output 并指定改善因子为 2,则产生输出的时刻为:

0, 1.25, 2.5, 3.75, 5, 6.75, 8.5, 9.25, 10

选择 Produce additional output 并指定[0:10]产生输出,则输出时刻为:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

也可能在其他附加时刻。这些依赖于变步长仿真器所选择的步长。

选择 Produce Specified Output Only 并指定[0:10]产生输出,输出时刻为:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

5.3.2 Workspace I/O 选项设置

通过对该页的设置,可以将仿真结果输出到工作空间的变量,也可从工作空间得到输入和初始状态。在 Simulation Parameters 对话框内,选择 Workspace I/O 标号,页面出现如图 4.6 所示。

1. 从基本工作空间载入输入

在仿真运行时,Simulink 能把输入从一个模型基本工作空间加载到模型的最高层的输入端口。定义这一项时,选中 Workspace I/O 上 Load from workspace 区域的 Input 复选框,然后在旁边的编辑框内输入一个外部输入,然后选中 Apply。外部输入可以采用下列任一种格式:

(1) 外部输入矩阵(External Input Matrix)。外部输入矩阵的第一列必须为递增的时间矢量,其余列指定输入值。特别地,每一列数代表不同的 Inport 模块信号(依次)输入,每一行

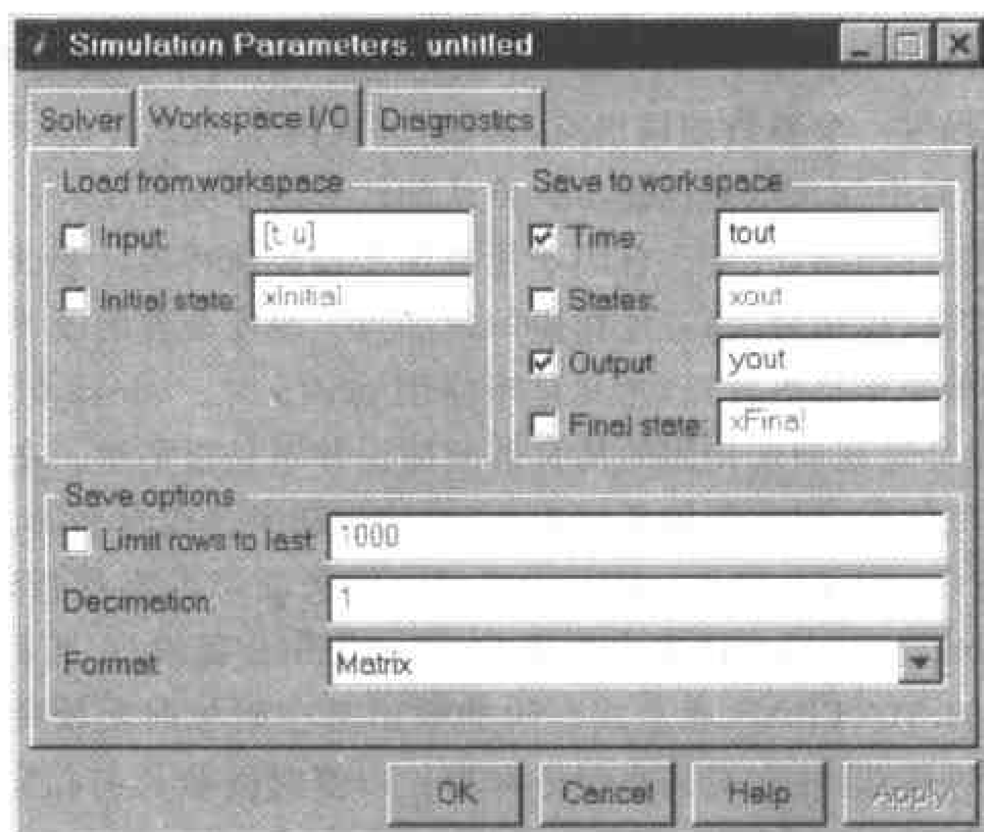


图 5.6 Workspace I/O 页对话框

是相应的时间点的输入值。如果 Interpolate data 项被选中作为相应的输入(参看 Inport 模块的 Interpolate data 选项),则 Simulink 在必要时对输入进行线性内插或者外插计算。

输入矩阵的总列数必须等于 $n+1$,其中 n 为输入到模型的信号总数。如果在基本工作空间定义了变量 t 和 u,就不需要为此模型输入外部输入要求。因为默认的外部输入要求为[t, u]。举例来说,假定某一个模型有两个输入端口,其中一个接受两个信号,另一个接受一个信号。又假定基本工作空间定义 t 和 u 如下:

```
>>t = (0:0.1:1);
>>u = [sin(t), cos(t), 4 * cos(t)];
```

然后,指定模型的外部输入,只需选中模型的外部输入复选框即可。

(2) 包含时间的结构(Structure with time)。Simulink 从工作空间以结构的形式读取数据,其结构的名字在 Input 文本编辑框内指定。该输入结构必须有两个最高层结构:time 和 signals。time 结构包含一个列矢量,表示仿真时间。signals 结构包含一个子结构数组,每个子结构对应模型的一个输入端口,每个子结构下必须包含字段 values。values 字段包含一个列矢量,是相应输入端口的输入数据。

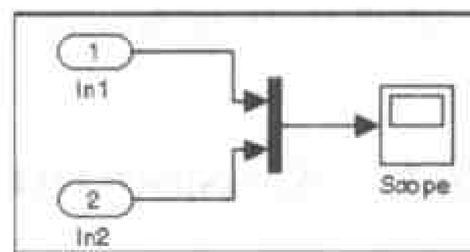


图 5.7 两个输入端口的模型

例如,考虑图 5.7 的模型,它有两个输入。

假定在基本工作空间已定义了一个输入模型,输入矢量 a,如下:

```
>>a.time = (0:0.1:1)';
>>a.signals(1).values = sin(a.time);
>>a.signals(2).values = cos(a.time);
```

然后,指定 a 为模型的外部输入,选中 Input 复选框,并在旁边的文本框内输入 a。

编者提示:Simulink 能以包含时间的结构读取仿真数据并保存到工作空间。详见 Inport 模块的 Structure with time。

(3) 不包含时间的结构(Structure)。结构格式与包含时间的结构格式相同,只是 time 域为空。在前面的例子中,可以制定时间域为

```
>>a.time = [   ]
```

在这个例子中,Simulink 在第一个时间步读取输入值数组的第一个元素,第二个时间步读取输入值数组的第二个元素,以此类推。

编者提示:Simulink 能以 Structure 格式读回保存到工作空间的仿真数据。更多信息,参看第 7 章"Inport 模块"中的 Structure。

(4) Per-Port 结构(Per-Port Structures)。本格式由独立的包含时间的结构和不包含时间的结构组成,每个输入端口对应一个结构。每个端口的输入数据结构只有一个 signals 域。要定义此选项,在 Input 旁边的文本编辑框输入各个结构的名字,中间用逗号分隔,如列表 in1,in2,...,inN,其中 in1 是模型第一个端口的数据,in2 是第二个端口的数据,以此类推。

(5) 外部输入时间表达式(External Input Time Expression)。时间表达式可以是任何 MATLAB 表达式,用来计算在长度上等于输入到模型输入端口信号数的行矢量。例如,假定一个模型有一个矢量端口接受两个信号。又假定:timefcn 是一个自定义的函数,返回有两个元素长的行矢量。下面就是该模型有效的输入时间表达式:

```
'[ 3 * sin(t), cos(2 * t)] '
'4 * timefcn(w * t) + 7'
```

Simulink 在各个仿真步计算该表达式,把结果值引入模型的端口。

编者提示:Simulink 在运行该仿真时定义变量 t。也可以省略变量的函数表达式中的时间变量,如,Simulink 将表达式 sin 解释为 sin(t)。

2. 保存输出到工作空间的数据

可以通过在本对话框页的 Save to Workspace 区域选择 Time,States 和/或 Output 复选框,来指定返回变量。指定返回变量,令 Simulink 将时间、状态和输出曲线(同选定数)的值写入到工作空间。

在复选框右边的编辑框内指定变量名可设定不同的变量。如要将输出写到多个变量中,须以逗号分隔各个变量名。Simulink 用一个矢量保存仿真时间,该矢量名在 Save to Workspace 区域中指定。

编者提示:Simulink 以模型的基准采样率将输出保存到工作空间中。如果要以不同的采样率保存输出,可以使用 To Workspace 模块。

保存选项可以指定输出保存量的格式和限制。模型状态和输出格式选项有:

(1) 矩阵(Matrix)。Simulink 在一个矩阵里保存模型的状态,矩阵名字在 Save to Workspace 区域指定(如,xout)。状态矩阵的每一列对应一个模型状态,每一行对应一个特定时间的各个状态。模型输出矩阵的名在 Save to Workspace 区域指定(如,yout)。矩阵的每一列对应一个模型输出,每一行对应一个特定时间的各个输出。

(2) 包含时间的结构。Simulink 以结构格式保存模型的输出,其结构名在 Save to Workspace 选项内指定(如,yout)。该结构有两个顶级结构:time 和 signals。Time 结构包含一个仿真时间矢量。Signals 结构包含一个子结构数组,每一子结构对应一个输出端口。每个子结

构又含有三个字段：values，label 和 blockName。Values 字段包含一个对应于各个端口的输出矢量。Label 字段指定连接到输出端口的信号标签。BlockName 字段指定输出端口的名字。Simulink 在与模型输出结构有同样组成的结构里保存该模型的状态。

(3) 不包含时间的结构。本格式与上面所述的包含时间的结构基本相同，只是不在 Time 域中保存仿真时间。

(4) Per-Port 结构。本格式由独立的包含时间的结构和不包含时间的结构组成，每个输出端口对应一个结构。每个输出数据结构只有一个 signals 域。要定义这一项，在 Output 文本编辑框内输入每个结构的名字，以逗号分隔，如 out1，out2，...，outN，其中 out1 是模型第一个端口的数据，out2 是第二端口的数据，以此类推。

要为保存的数据的最大行数制定一个界限，选中标有 Limit rows to last 的复选框，并指定保存行数。要引入抽取因子，即在几个数中选择一个，可在 Decimation 标签右边的编辑框内输入一个值。例如，输入 2，则表示保存每隔一个点的数据。

3. 载入和保存状态

初始化状态，即系统在仿真开始时候的状态，一般是在模块中设置。也可以通过指定 States 区域，来覆盖在模块中设置的初始状态。

还可以保存仿真的最后状态，将其应用到另一个仿真。这种情况在实际应用中是经常遇见的。在 Workspace I/O 选项的 Save options 子项下通过选择不同的格式(Format)保存状态。如果选中包含时间的结构或不包含时间的结构，则保存格式如下：

• 包含时间的结构(Structure With Time)。Simulink 以结构的格式保存模型的状态，其结构名在 Save to Workspace 区域的 Final State 编辑框中指定(如，xFinal)。该结构有两个最高级字段：time 和 signals。Time 字段包含一个仿真时间矢量。Signals 字段包含一个子结构数组，每一子结构对应一个具有状态的模块。每个子结构又含有三个字段：values，label 和 blockName。values 包含一个对应于各个模块的状态矢量。label 可以是 Cstate(连续状态)，或 Dstate_n，其中 n 可以是 1，2，3，…直到设定的相应模块离散状态的最大数。BlockName 字段指定由此结构元素代表的模块名。

• 不包含时间的结构。本格式与上面所述的包含时间的结构基本相同，只是不在 Time 字段中保存仿真时间。

(1) 载入状态，可以通过选择 Initial State 复选框来指定包含初始状态值的变量名。该变量可以是一个矩阵，或者是与保存最后状态的格式相同的结构体。使用包含时间的结构或者不包含时间的结构，可以将当前部分的初始状态设置为所保存的前一个部分的状态。如果复选框没有选中，或状态向量是空的([ ])，则 Simulink 使用由模块定义的初始状态。

(2) 保存最后状态(仿真终止时的状态值)，可以选中 Final State 复选框并在旁边的编辑框内输入一个变量。

(3) 当模型有多个状态时，就要为有多个状态的模型指定初始条件，并需要确定状态的次序。使用下面命令可以指定初始条件并确定状态的次序：

```
>>[sizes, x0, xstord] = sys([ ], [ ], [ ], 0)
```

其中 sys 是模型名。该命令返回：

① sizes，指示该模型特征的矢量。只有前两个元素应用于初始条件：sizes(1)是连续状态的数目，sizes(2)是离散状态的数目。

② x0,模块的初始条件。

③ xstord,一个字符串矩阵,包含模型中所有具有状态的模块的路径名。在 xstord 和 x0 矢量中的模块次序是相同的。

例如,下面的表达式可以获得 vdp 模型的初始条件的值和状态次序(该例子只显示了 sizes(1)的值、连续状态的数目,sizes(2)、离散状态的数目):

```
>>[sizes, x0, xstord] = vdp([ ], [ ], [ ], 0)
sizes =
     2
     0
     2
     0
     0
     0
     1
x0 =
     2
     0
xstord =
     'vdp/x1'
     'vdp/x2'
```

5.3.3 Diagnostics(诊断)选项设置

在 Simulation Parameters 对话框的 Diagnostics 选项,用于设置系统在仿真过程中遇到一些事件或状态时应做出的反应。该选项的对话框如图 5.8 所示。

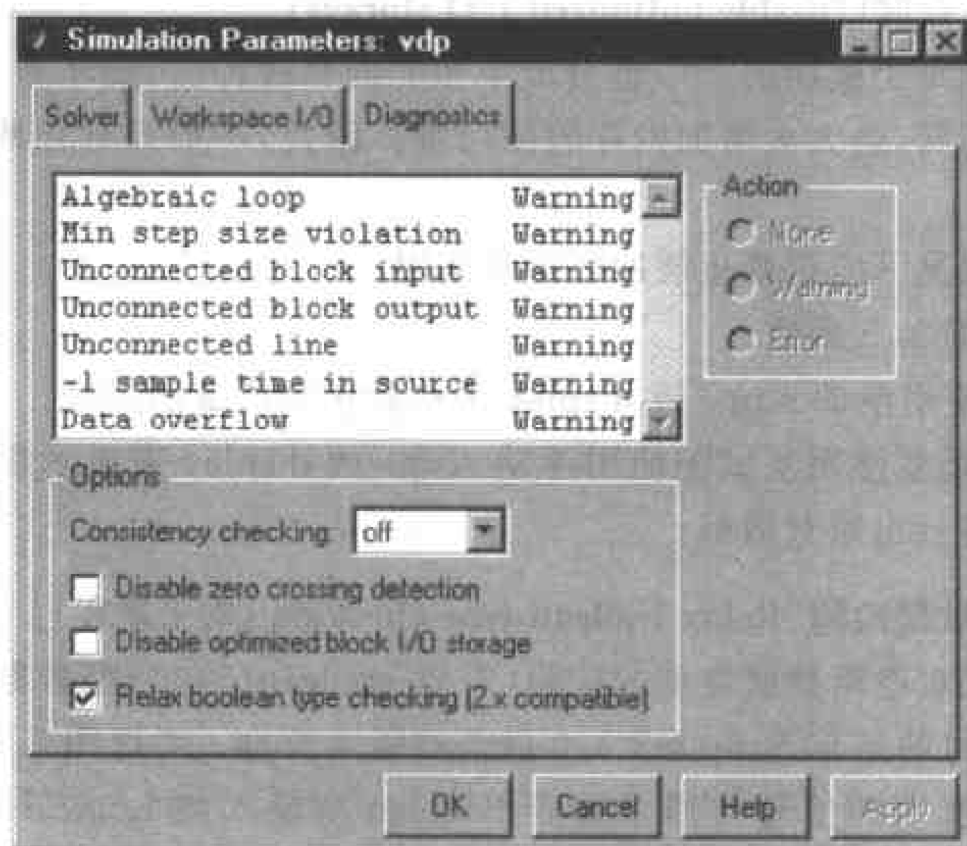


图 5.8 Diagnostics 对话框

对于每个事件类型，可以选择无信息、警告信息，或者出错信息。警告信息并不停止仿真，但出错信息将停止仿真。包括以下选项：

1. 一致性检验(Consistency checking)

一致性检验是一个调试工具，用来检验 Simulink ODE 仿真器所做的某些假设。它的主要用途是保证系统函数(S-functions)和 Simulink 内置模块服从相同的规则。由于一致性检验会明显导致系统性能下降(运行速度下降近 40%)，一般情况下设置为 off。使用一致性检验不仅可以验证系统函数，也有助于对意外仿真结果原因的分析。

为了提高积分效率，Simulink 从一个时间步保存(缓存)某些值用于下一个时间步。例如，一般情况下，一个时间步结尾时的微分可以在下一个时间步的开始重新利用。利用这一点，仿真器避免了许多多余的微分计算。

一致性检验的另一个目标是确保模块在给定的 t(时间)值输出不变。这对于刚性仿真器(ode23s 和 ode15s)是很重要的，因为在计算 Jacobian 系数时，模块的输出函数会在同一 t 值被调用很多次。

当一致性检验启用时，Simulink 再次计算适当的值，并与缓存值比较。如果不同，则发生一致性错误。Simulink 比较计算的量有：

- 输出；
- 过零；
- 导数；
- 状态。

2. 禁止过零检测(Disable zero crossing detection)

用户可以禁止一个仿真中的过零检测，因为对于有过零的模型，禁止过零检测可以加速仿真，但是可能会使仿真的精度下降。

该选项仅对本身固有过零检测的模块禁止过零检测，而对 Hit Crossing 模块，并不禁止。

3. 禁止优化 I/O 存储(Disable optimized I/O storage)

选中此项，将会给每个模块的 I/O 值分配单独的内存缓冲区，而不重复使用内存缓冲区。这样，对于大模型的仿真，会大大增加内存的使用量。因此，一般只有在调试模型时才选中此项。

特别地，在下列情况下应该禁止缓冲区重复使用：

- 调试一个 C-MEX S 函数。
- 调试模型时，使用浮动 scope 或 display 来检验正在调试的信号。

如果缓冲区可以重复使用且试图使用浮动 scope 或 display 来显示缓冲区被重复使用的信号时，Simulink 将显示出错对话框。

4. 非严格布尔型数据检验[Relax boolean type checking (2.x 兼容)]

选中本项，会放宽对逻辑数据类型的检验，需要 boolean(布尔)型数据输入的模块可以接受双精度型数据。其目的是与 Simulink 3.0 以前的版本兼容。考虑图 5.9 的例子：

这个模型把双精度型的信号与通常需要 boolean 型输入的 Logical Operator(逻辑操作符)模块连接在一起。当 Relax boolean type checking 被选中时，模型在运行时不出现错误。

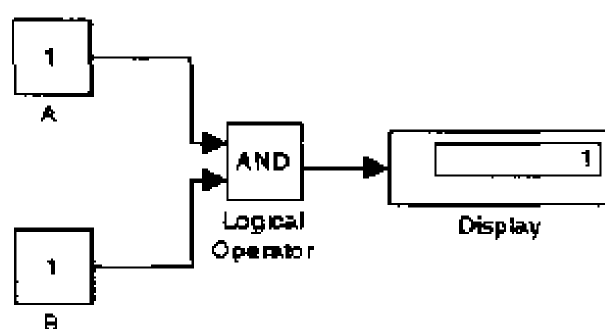


图 5.9　一个逻辑模型

5.4　提高仿真性能和精度

仿真的性能和精度受许多因素的影响，包括模型设计和仿真参数选择。本节介绍一些影响仿真的原因和解决的方法。

一般情况下，仿真器能在默认参数值时精确、有效地处理大多数的模型仿真。然而，如果适当调整模型的仿真器和仿真参数，会产生更好的结果。也就是说，如果知道该模型的性能信息并提供给仿真器，仿真的结果也会更好。

5.4.1　提高仿真速度

有许多因素都会引起仿真速度减慢。如：

(1) 模型包含 MATLAB Fcn 模块。当模型包含 MATLAB Fcn 模块时，在每一步，都会调用 MATLAB 解释程序(器)，这就大大地减慢了仿真。因此，尽可能地使用内置 Fcn 模块。

(2) 模型中包含 M 文件 S 函数。M 文件 S 函数也在每一步引起 MATLAB 解释器的调用。可以考虑将 S 函数转换为一个子系统或者一个 C-MEX S 函数。

(3) 模型包含 Memory 模块。使用 Memory 模块会使变阶仿真器(ode15s 和 ode113)在每一步将阶数重复设置为 1。

(4) 最大步长太小。如果用户改变了最大步长，必须使用默认值(auto)再一次运行该仿真。

(5) 精度要求太高。默认的相对容限(精度 0.1%)通常是足够的。对于有零状态的模型，如果绝对容限参数太小，接近零状态值会需要太多的步。参看上节中有关误差容限的内容。

(6) 时间刻度太大。减小时间间隔。

(7) 使用非刚性仿真器去解决刚性问题。尝试使用 ode15s 仿真器。

(8) 模型中使用的采样时间彼此间不成倍数关系。这样的混合采样时间会使仿真器采用过小的步长，以保证采样时间能精确地反映所有的采样时间，这导致运行速度降低。

(9) 模型中包含代数环。代数环的解法是在每步进行迭代运算。因此，大大降低了执行性能。更多的信息参见 6.6.1 节中有关代数环的内容。

(10)模型将一个 Random Number(随机数)模块反馈到一个 Integrator(积分)模块。对于连续系统，应使用 Sources 库中的 Band-Limited White Noise 模块。

5.4.2 提高仿真精度

(1) 检查仿真精度。在一个合理的时间段内运行仿真，然后，将相对误差容限(Relative Tolerance)减少到1e-4(默认为1e-3)或减小绝对误差容差，并再次运行仿真。比较两者的仿真结果。如果结果没有大的不同，可以确信该算法收敛。

(2) 如果仿真在开始时就漏过了关键的操作，则必须减少初始步长(Initial Step Size)，以确保仿真不“越过”这些关键操作。

(3) 如果仿真结果在时间上不稳定，则意味着：

① 用户系统可能是不稳定的。

② 如果用户使用的是ode15s，则需要限制最大阶为2(该仿真器为A级稳定时的最大阶)，或者尝试使用ode23s仿真器。

(4) 如果仿真结果不精确，则意味着：

① 对于含有接近零值状态的模型，如果绝对误差容限参数太大，仿真会在零值状态周围采用过少的操作步。减小该参数值或者在Integrator模块对话框内调整它的各自状态。

② 如果减少绝对误差容限并未有效地提高精度，则减少相对误差容限参数，来减小可接受误差和强迫使用小步长和多步操作。

5.5 在命令行输入命令运行仿真

在MATLAB命令窗口输入命令或用一个M文件都可以自动运行仿真。可以在命令行输入sim命令或set_param命令控制运行仿真。具体方法描述如下。

5.5.1 使用sim命令

完整的句法为：[t,x,y] = sim(model, timespan, options, ut);

命令中只需要model参数，命令中没有提供的参数将从Simulation Parameters对话框所设置的相应参数中读取。

sim命令的详细句法，参见sim命令。options参数提供附加的仿真参数，包括仿真器名和误差容限，其数据格式为结构体。可以使用simset命令在options结构中定义参数(详见本节simset命令)。

编者提示：这些命令经常是需要配合使用的。

5.5.2 使用set_param命令

可以使用set_param命令来开始、停止、暂停或继续一个仿真，或更新一个模块图。类似地，可以使用get_param命令来检查仿真的情况。此种用法的set_param命令格式为：

```
set_param('sys', 'SimulationCommand', 'cmd')
```

其中'sys'是系统名，'cmd'可以是'start'，'stop'，'pause'，'continue'或'update'。

此种用法的get_param的格式为：

```
get_param('sys', 'SimulationStatus')
```

Simulink返回'stopped'、'initializing'、'running'、'paused'、'terminating'和'external'(使用实时工作空间)。

5.5.3　sim 命令

(1) 命令用途：动态系统的仿真。

(2) 命令格式：[t,x,y] = sim(model,timespan,options,ut);

[t,x,y1, y2, ..., yn] = sim(model,timespan,options,ut);

(3) 命令描述：sim 命令对用 Simulink 模型表示的动态系统进行仿真。用户可以提供一个零([])矩阵替代等式右侧的任何一个变量，但第一个除外(模型名)。默认时，sim 命令使用零矩阵的变量。可以使用 sim 命令的 options 变量设置任选仿真参数。用这种方法设置的参数将覆盖由模型设置的参数。

如果是连续系统仿真，必须使用 simset 命令(见后)指定仿真器参数。该仿真器为纯离散模型，默认为 VariableStepDiscrete。

表 5.2 为参数及说明。

表 5.2　参数及说明

参数	说　明
t	返回该仿真时间矢量
x	返回仿真状态矩阵，由连续状态组成，其后是离散状态
y	返回仿真输出矩阵。每一列按端口顺序包含一根级 Outport 模块的输出。如果任何 Outport 模块有矢量输入，其输出到适当的列数
y1,...yn	对于一个有 n 个 Outport 模块的模型，每个 yi 返回相应根级 Outport 模块的输出
model	模块图表名
timespan	仿真开始和停止时间。指定为如下之一： tFinal 指定停止时间。其起始时间为 0 [tStart tFinal]指定开始和停止时间 [tStart OutputTimes tFinal]指定开始、停止时间和在 t 中返回的时间点。通常，t 将包含更多的时间点。OutputTimes 是相当于选择对话框中的 Produce additional output
option	由 simset 命令设置的仿真参数，其数据格式为结构体
ut	输入到顶级 Inport 模块的可选外部数据。Ut 可是一个指定每个仿真时间步的输入 U = UT(t)的 MATLAB 函数(以字符串表示)，可以是所有输入端口相对于时间的输入数据表，或者是由逗号分隔的列表 ut1, ut2, ...，每个对应一个具体的端口。所有端口的制表输入可以是 MATLAB 矩阵或结构的形式。单独端口的制表输入必须是结构的形式。参看本章“从基本工作空间载入输入”中对矩阵和结构输入格式的描述

(4) 应用举例。下面的命令仿真 Van der Pol 方程，使用 Simulink 的 vdp 模型。该命令使用所有默认参数

```
>>[t,x,y] = sim('vdp')
```

下面的命令仿真 Van der Pol 方程，使用与 vdp 模型相关联的参数，只定义了 Refine 参数。

```
>>[t,x,y] = sim('vdp', [ ], simset('Refine',2));
```

下面的命令仿真 Van der Pol 方程 1 000s，保存返回变量的最后 100 行。仿真只输出 t 值和 y 值，但在变量 xFinal 中保存最后的状态矢量。

```
>>[t,x,y] = sim('vdp', 1000, simset('MaxRows', 100,'OutputVariables', 'ty', 'FinalStateName', 'xFinal'));
```

(5) 相关命令：simset，simget 命令。

5.5.4 simset 命令

(1) 命令用途：创建或者编辑 sim 命令的仿真参数和仿真器属性。

(2) 命令格式：

```
options = simset(property, value, ...);
options = simset(old_opstruct, property, value, ...);
options = simset(old_opstruct, new_opstruct);
simset
```

(3) 命令描述：该命令创建一种称为 options 的结构，其中命名的仿真参数和仿真器属性有指定值，所有未指定的参数和属性采用它们的默认值。仅需输入足够的前导字符来惟一地标识该参数或者属性即可。参数和属性忽略空格。

options = simset(property, value, ...)命令把命名属性(property)赋值给(value)，并且把结果存储到 options 中。

options = simset(old_opstruct, property, value, ...)命令改变现有结构 old_opstruct 的命名属性 property，并赋值给 value。

options = simset(old_opstruct, new_opstruct)命令把两个现有的选项结构 old_opstruct 和 new_opstruct 合并为 options，在 new_opstruct 中定义的所有特性将改写 old_opstruct 中的特性。

用不带参数的 simset 指令可以显示所有的属性以及它们的可能值。

编者提示：不能使用 get_param 和 set_param 命令获得或者设置这些属性和参数。

(4) 命令参数及说明：

- AbsTol

绝对误差容限，正标量(默认 1e-6)，本标量适用于状态矢量的全部元素。但 AbsTol 只适用于变步仿真器。

- Decimation

输出变量的抽样率，正整数(默认 1)，抽样因子适用于返回变量 t，x 和 y。当取值 1 时，返回每个记录时间点数据，值为 2 时返回每隔一点的记录时间点数据，以此类推。

- DstWorkspace

变量的赋值之处，默认值为当前(current)。本属性指定工作空间，该工作空间用来对已定义为返回变量或输出变量的变量进行赋值。

- FinalStateName

最终状态变量名，字符串(默认'')。本属性指定 Simulink 为用来保存仿真末尾状态的变

量名。

- FixedStep

固定步长，正标量。本属性只适用于固定步长仿真器。若该模型包含离散成分，默认值是基本采样时间；否则，默认值取仿真周期的1/50。

- InitialState

初始状态，矢量{默认零矩阵[]}。该初始状态矢量由连续状态和紧跟其后的离散状态组成。InitialState取代模型指定的初始状态。默认情况下用一零矩阵引入使用模型定义的初始状态值。

- InitialStep

建议初始步长，正标量{默认自动auto}。本属性只适用于变步长仿真器。首先仿真器会尝试一个InitialStep步长。默认时，该仿真器会自动地确定一个初始步长。

- MaxOrder

ode15s的最大阶，1 | 2 | 3 | 4 | {默认5}。本属性只适用于ode15s。

- MaxRows

输出限定行数，非零整数{默认0}。本属性用上一次记录时间点的MaxRows数据来限制t，x，和y返回的行数。如果指定为0(即默认值)，表示不限定。

- MaxStep

步长上限，正标量{默认自动auto}。本属性只适用于变步长仿真器，并预先设定为仿真时间的1/50。

- OutputPoints

确定输出点，{默认指定}|全部。本属性设定为specified，仿真器只在timespan指定的时间点输出t，x和y。当设定为all时，仿真器所用的时间步内的t，x和y也会被输出。

- OutputVariables

设定输出变量，{默认txy} | tx | ty | xy | t | x | y。如果该属性字符串中缺少't'，'x'或'y'，仿真器就在对应输出t，x或y中产生一个零矩阵。

- Refine

输出改善因子，正整数(默认1)。本属性通过指定该因子增加输出点数目，以平滑输出。本因子只适用于变步长仿真器。如果输出时间被指定，它会被忽略。

- RelTol

相对误差容限，正标量{默认1e-3}。本属性适用于状态矢量的全部元素。每一积分步内的错误估计满足：

```
e(i) <= max(RelTol * abs(x(i)),AbsTol(i))
```

本属性只适用于变步长仿真器，并预先设定为1e-3，其精度在0.1%以内。

- Solver

本属性指定算法。

VariableStepDiscrete | ode45 | ode23 | ode113 | ode15s | ode23s | FixedStepDiscrete | ode5 | ode4 | ode3 | ode2 | ode1

- SrcWorkspace

{ } | current | parent

表达式的计算之处{默认基本 base }|当前|上级。本属性指定用来计算在模型中定义的 MATLAB 表达式的工作空间。

- Trace

跟踪工具，'minstep'，'siminfo'，'compile' {''}。本属性激活仿真跟踪工具(指定一个或多个，由逗号分隔的列表)：

①跟踪标记'minstep'指定当算法突然改变，使得变步长仿真器不能找到满足误差容限的步长时，仿真停止。默认时，Simulink 发送一个警告信息并继续该仿真。

②跟踪标记'siminfo'在仿真一开始的时候提供有效仿真参数的摘要信息。

③跟踪标记'compile'显示一模型的编译状态。

- ZeroCross

激活或禁止过零检测定位，{默认 on }| off。本属性只适用于变步长仿真器。如果设置为 off，变步长仿真器将不对固有过零检测的模块进行过零检测。仿真器仅调整它们的步长来满足误差容限。

(5) 应用举例：

本命令创建名为 myopts 的选项结构，它定义 MaxRows 和 Refine 参数值，对其他参数使用默认值：

```
>>myopts = simset('MaxRows', 100, 'Refine', 2);
```

下面的命令仿真 vdp 模型 10 s 并使用在 myopts 中定义的参数：

```
>>[t,x,y ] = sim ( 'vdp',10,myopts );
```

(6) 相关命令：sim，simget 命令。

5.5.5 simget 命令

(1) 命令用途：得到选项结构属性和参数。

(2) 命令格式：struct = simget(model)

value = simget(model, property)

(3) 命令描述：simget 命令得到指定模型的仿真参数和仿真器属性值。如果一个参数或属性是用变量名定义的，simget 则返回该变量值，而不是变量名。如果该变量在工作空间中不存在，Simulink 提示一出错信息。

struct = simget(model)　返回指定模型的当前结构。该结构使用 sim 和 simset 命令定义。

value = simget(model, property)　从模型提取指定仿真参数或仿真器属性。

value = simget (OptionStructure,property)　从 OptionStructure 提取指定仿真参数或仿真器属性，如果该值没有在结构中指定，返回一零矩阵。Property(属性)可以是一单元数组，包含参数列表和属性名。如果使用的是单元数组，输出也是单元数组。

用户只需输入属性名的几个前导字符，便于惟一识别即可。属性名忽略格的作用。

(4) 应用举例：

下面命令为 vdp 模型检索选项结构

```
>>options = simget('vdp');
```

下面命令为 vdp 模型检索 Refine 属性值

```
>>refine = simget('vdp', 'Refine');
```

(5) 相关命令：sim，simset 命令。

5.6　仿真结果的分析

Simulink 提供了一些用于观察、记录输出的模块和一些用于模型参量分析的函数。

显示输出有下面三种途径：

- 将信号输入 Scope 模块或 XY Graph 模块；
- 将输出数据写到返回变量，并使用 MATLAB 显示图形的命令；
- 使用 To Workspace 模块将输出数据写到工作空间，并使用 MATLAB 显示图形的命令显示波形。

5.6.1　使用 Scope 模块观察输出信号

在仿真过程中，可以将输出波形显示在 Scope 模块上。在图 5.10 的模型中就使用了 Scope 模块。

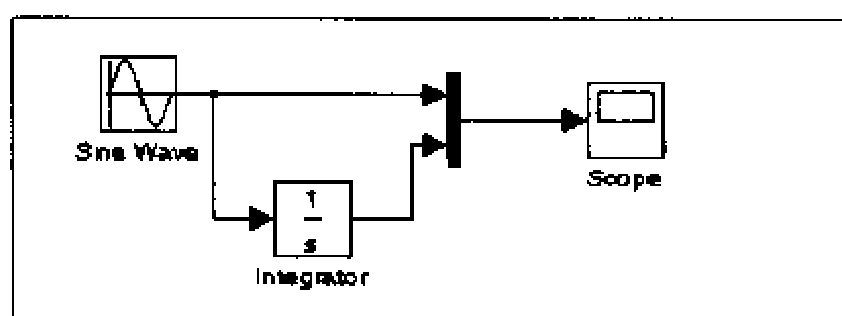


图 5.10　包含 Scope 模块的模型

Scope 上显示了输出波形，Scope 模块可以放大感兴趣的图形区域，也可以将数据保存到工作空间。XY Graph 模块还可以显示两个波形的关系。

这些模块将在第 7 章中详细描述。

5.6.2　使用返回变量

通过返回时间及输出历史记录，用户可以使用 MATLAB 图形显示命令来绘制和注释输出信号的轨迹。

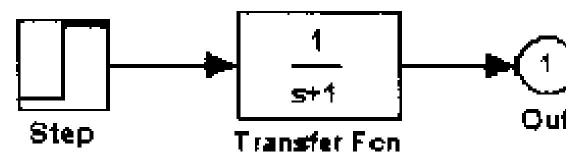


图 5.11　包含 Outport 模块的模型

如图 5.11 所示，标签为 Out 的模块是 Signals & Systems 库中的 Outport 模块。积分仿真器返回输出信号轨迹，yout。

也可以从 Simulation 菜单中运行此模型的仿真，需通过在 Simulation Parameters 对话框中的 Workspace I/O 页上指定时间变量、输出变量以及状态数变量。这样，就可以使用下面的命令显示结果的图形：

```
>>plot(tout,yout)
```

5.6.3　使用 To Workspace 模块

To Workspace 模块可以将输出信号轨迹送到 MATLAB 工作空间。下面举例说明该模

块的使用方法，如图 5.12 所示。

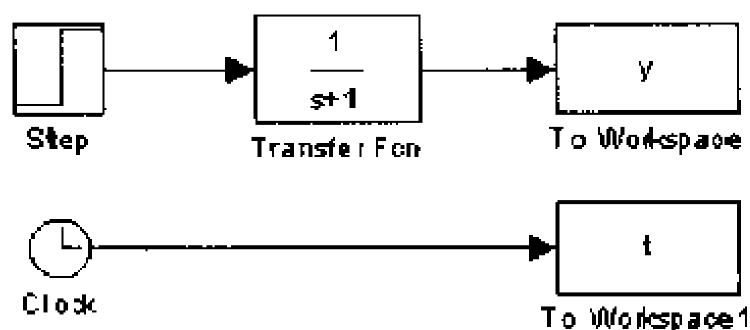


图 5.12 To Workspace 模块的使用范例

当仿真完成时，变量 y 和 t 出现在工作区。时间矢量的储存是通过将一个 Clock 模块的输出送入 To Workspace 模块来实现的。时间矢量还可以通过在 Simulation Parameters 对话框的 Workspace I/O 上输入一个时间变量名获得，或使用 sim 命令返回时间矢量。

To Workspace 模块能接受矢量输入，且各个输入元素的轨迹以列矢量的方式储存在工作空间结果变量中。

5.7 线性化与线性分析

Simulink 提供了 linmod 和 dlinmod 函数，以状态空间矩阵 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$ 的形式来提取线性模型。状态方程矩阵的输入输出线性关系描述如下：

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu}$$

$$\boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du}$$

其中，$\boldsymbol{x}$、$\boldsymbol{u}$ 和 $\boldsymbol{y}$ 分别是状态、输入和输出矢量。

5.7.1 线性模型

请看图 5.13 所示的例子，该模型名为 lmod。

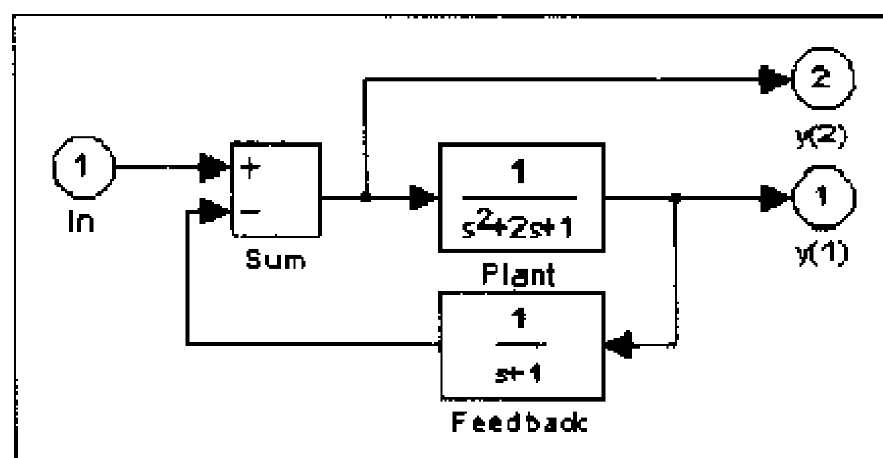


图 5.13 线性模型

为了提取该 Simulink 系统的线性模型，输入命令：

```
>>[A,B,C,D] = linmod('lmod')
A =
      -2  -1  -1
```

```
    1 0 0
    0 1 1
B =
    1
    0
    0
C =
    0 1 0
    0 0 1
D =
    0
    1
```

编者提示：模型的输入和输出必须使用 Signals & Systems 库中的 Inport 和 Outport 模块定义。源和接收模块并不作为输入和输出。通过 Sum 模块，Inport 模块可以与源模块连接。

一旦数据成为状态方程的格式，或者转换为 LTI 对象，就可以应用控制系统工具箱（Control System Toolbox）中所提供的函数做进一步的分析：

(1) 转换为 LTI 对象：

```
>>sys = ss(A,B,C,D);
```

(2) 波特图（相-频和幅-频特性）：

```
>>bode(A,B,C,D) 或 bode(sys)
```

(3) 线性化时间响应：

```
>>step(A,B,C,D) 或 step(sys)
>>impulse(A,B,C,D) 或 impulse(sys)
>>lsim(A,B,C,D,u,t) 或 lsim(sys,u,t)
```

控制系统工具箱和鲁棒控制工具箱（Robust Control Toolbox）中的一些其他函数都可以用于线性控制系统的设计。

5.7.2 非线性模型

当模型是非线性时，就应选择一个运算点来提取线性模型。非线性模型对发生在提取模型点的扰动大小也是敏感的。必须在各种舍入与截位误差之间选择一种折衷方案。Linmod 命令中的额外参数用于指定运算点和扰动点：

```
>>[A,B,C,D] = linmod('sys', x, u, pert, xpert, upert)
```

5.7.3 离散系统或者混合连续离散系统

此时，应使用函数 dlinmod 来线性化。除了从右手数第二个参数一定要包含线性化时必需的采样时间外，该函数和函数 linmod 有一样的调用句法。详细资料请参见 5.9 节中的 linfun 函数。

使用 linmod 函数对包含 Derivative(微分)或 Transport Delay(传输延迟)模块的模型进行线性化可能有些麻烦。在线性化之前，用特别设计的模块替换这些模块可以避免出现问题。这些特殊模块在 Linearization 子库中的 Simulink Extras 库中，可以通过点击 Blocksets & Toolboxes 图标来打开 Extras 库。

(1) 对于 Derivative 模块，须使用转换的(Switched)微分来线性化。

(2) 对于 Transport Delay 模块，需使用转换的(Switched)传输延迟来线性化。(使用本模块要求有 Control System(控制系统)工具箱。)

当使用一个 Derivative 模块时，还可以将该微分模块合并到其他模块中。例如，如果一个微分模块与一个传递函数模块串联，最好是用单个传递函数(Transfer Fcn)模块来实现(虽然并不是总能实现)，即：

$$\frac{s}{s+a}$$

在图 5.14 中，左边的模块可以用右边的模块替换。

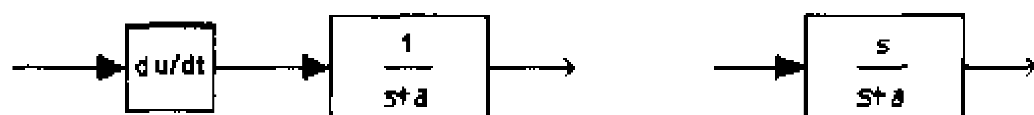


图 5.14 两个等价模型

5.8 平衡点的确定

Simulink 的 trim 函数可用来确定稳态的平衡点。请看图 5.15 的名为 lmod 的模型范例。

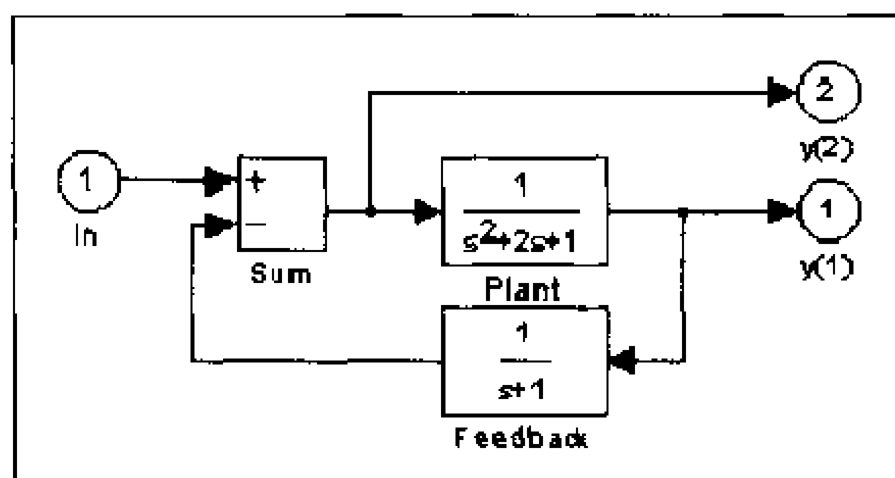


图 5.15 lmod 模型

使用 trim 函数可以找到输入值和使两个输出同时为 1 的状态值。

步骤 1:赋予状态变量(x)和输入值(u)初始的试验值，然后设定输出值(y)：

```
>>x = [ 0; 0; 0 ];
>>u = 0;
>>y = [ 1; 1 ];
```

步骤 2:使用索引变量指示哪些变量是固定的，哪些变量是可变的：

```
>>ix = [ ];                % 不固定任何状态
```

```
>>iu = [ ];              % 不固定任何输入
>>iy = [ 1;2 ];          % 固定输出 1 和输出 2
```

步骤 3:调用 trim 函数返回解(由于舍入误差,结果有可能有所不同)。

```
[x,u,y,dx ] = trim( 'lmod',x,u,y,ix,iu,iy )
x =
   0.0000
   1.0000
   1.0000
u =
   2
y =
   1.0000
   1.0000
dx =
   1.0E-015 *
   0.2220
   0.0227
   0.3331
```

编者提示:对于平衡点问题,有可能会没有结果,如果是这种情况,在第一次尝试设置及微分为零后,trim 函数返回一个使与理想值之间的最大偏差减到最小的值。关于 trim 的句法描述详见本章 trim 函数。

5.9 linfun 函数

5.9.1 命令用途

在一运算点周围提取系统的线性状态空间模型。

5.9.2 命令格式

```
[A,B,C,D] = linfun('sys')
[A,B,C,D] = linfun('sys', x, u)
[A,B,C,D] = linfun('sys', x, u, pert)
[A,B,C,D] = linfun('sys', x, u, pert, xpert, upert)
```

5.9.3 命令参数

表 5.3 参数及说明

参 数	说 明
linfun	Linmoa,dlinmod 或 linmod2
sys	Simulink 的系统名,从它来提取线性模型

续表 5.3

参　数	说　　明
x 和 u	状态和输入矢量。如果被指定，则将运算点设置在线性模型提取的地方
xper 和 upert	用于 *x* 和 *u* 的可选的标量扰动因子。如果没有指定，默认值为 1e-5 可选的矢量，用于直接设置各自状态和输入的扰动级别。如果被指定，则 pert 参数忽略
xpert 和 upert	第 *i* 个状态 *x* 扰动为：*x*(*i*)+xpert(*i*) 第 *j* 个输入 *u* 扰动为：*u*(*j*)−upert(*j*)

5.9.4 命令描述

linmod 从描述为 Simulink 模型的常微分方程的系统获得线性模型，以状态空间 ***A***，***B***，***C***，***D*** 的形式返回线性模型，描述了线性化后的输入-输出关系：

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

模型的输入和输出必须使用 Inport 和 Outport 模块。

[A,B,C,D] = linmod('sys')命令获得运算点周围的 sys 线性模型，其状态变量 *x* 和输入 *u* 为零。

Linmod 给运算点周围的状态加一个扰动以确定状态微分的变化速率和输出（Jacobians），其结果用来计算状态方程矩阵。给每个状态 *x*(*i*)加扰动：

$$x(i) + \Delta(i)$$

其中

$$\Delta(i) = \delta(1 + |x(i)|)$$

同样给第 *j* 个输入加一个扰动：

$$u(j) + \Delta(j)$$

其中

$$\Delta(j) = \delta(1 + |u(j)|)$$

1. 离散时间系统的线性化

函数 dlinmod 能够以任何给定的采样速率对离散系统、多采样速率系统和连续与离散混叠系统进行线性化处理。dlinmod 与 linmod 使用相同的调用句法，但是要在第二个表达式中插入执行线性化的采样时间，例如命令

```
>>[Ad,Bd,Cd,Dd] = dlinmod('sys', Ts, x, u);
```

产生一个采样时间为 Ts，运算点由状态矢量 x 和输入矢量 u 给定的、离散的状态方程模型。要获得离散系统的一个近似的连续模型，把 Ts 设定为 0。

由于组成系统的模块有线性的、多采样率的、离散和连续模块，dlinmod 在转换采样时间 Ts 下，产生的线性模型有完全相同的频率和响应时间（输入常量时），但必须满足以下条件：

- Ts 是系统中所有的采样时间的整数倍。
- Ts 不小于系统中最小的采样时间。

• 该系统是稳定的。

在有些情况下,虽然这些条件不满足,但也有可能会得到有效的线性模型。

计算线性矩阵 Ad 的特征值以指示该系统是稳定的。当系统稳定时,Ts>0 且特征值全部在单位圆内,则如下描述:

```
all(abs(eig(Ad))) < 1
```

同样地,当系统稳定时,如果 Ts=0 且特征值在左半边平面内,则如下描述:

```
all(real(eig(Ad))) < 0
```

当系统不稳定时,且采样时间不是其他采样时间的整数倍,dlinmod 产生 Ad 和 Bd 矩阵,该矩阵可能是复数阵。在这样的情况下,Ad 矩阵的特征值仍然提供系统稳定性的指示。

可以使用 dlinmod 将系统的采样时间转换为其他值,或将线性离散系统转换为连续系统,反之亦然。

使用 bode 命令可以得到一个连续或离散系统频率响应。

2. 高级线性化

linmod2 提供了高级线性化的格式。这个程序运行时间比 linmod 的长,但可以产生更精确的结果。

linmod2 的调用句法类似于 linmod,但功能不同。例如,linmod2('sys',x,u)产生一个线性模型,与 linmod 相同;然而,各个状态方程矩阵中各元素的扰动级别是逐个设置以使舍入和舍位误差减到最小。linmod2 寻求平衡舍入误差(由小的扰动级别引起,产生的误差与有限精度方法有关)及舍位误差(由大的扰动级别引起,它使分段线性近似值无效)。

式[A,B,C,D] = linmod2('sys',x,u,pert)中的变量 pert 指示能够使用的最小的扰动级别,默认值是 1e-8。linmod2 有检测断点和产生警告信息的优点,如:

```
Warning: discontinuity detected at A(2,3)
```

当这样的一个警告发生时,尝试不同的运算点,来获得线性模型。

式[A,B,C,D] = linmod2('sys',x,u,pert,Apert,Bpert,Cpert,Dpert)中的变量 Apert,Bpert,Cpert 和 Dpert 是用来设置各个状态和输入组合的扰动级别的矩阵,因此,Apert 的第 ij 个元素是与获得 A 矩阵的第 ij 个元素的相关的扰动级。用下式返回默认扰动大小:

```
>> [A,B,C,D,Apert,Bpert,Cpert,Dpert] = linmod2('sys', x, u)
```

编者提示:默认时,系统时间设置为零。对于依赖于时间的系统,可以把变量 pert 设置为有两个元素的矢量,第二个元素用来设置 t 的值,即获得线性模型的点。

当正在线性化的模型本身是一个线性模型时,不存在舍位误差问题。因此,可以把扰动级设置为任何期望的值。相对比较高的值更为可取,因为这有益于减少四舍五入误差。运算点的选择不影响线性模型的获得。

从非线性模型到线性模型的状态排序是不变的。对于 Simulink 系统,一个包含与各个状态相关的模块名的字符串变量可以通过下面命令获得:

```
>>[sizes,x0,xstring] = sys
```

其中,xstring 是字符串矢量,它的第 i 行是与第 i 个状态相关的模块名。输入和输出在模型图中按顺序编号。

对于单输入多输出的系统,用户可以使用例程 ss2tf 将其转换为传递函数形式,或使用 ss2zp 转化为零-极-增益形式。还可以使用 ss 将该线性模型转换为 LTI 对象。这个函数产生

一个状态方程形式的LTI对象，可以进一步转化为传递函数，或使用tf或zpk，转化为零-极-增益形式。

5.10 trim函数

5.10.1 命令用途

发现动态系统的trim点。

5.10.2 命令格式

```
[x,u,y,dx] = trim('sys')
[x,u,y,dx] = trim('sys',x0,u0,y0)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx)
[x,u,y,dx] = trim('sys',x0,u0.y0,ix,iu,iy,dx0,idx,options)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx,options,t)
[x,u,y,dx,options] = trim('sys',...)
```

5.10.3 命令描述

trim点，又称平衡点，是处于稳定状态的动态系统的参数空间的点。举例来说，飞机的trim点是使飞机平直飞行的一系列控制的装置。用数学方法描述，trim点是系统状态导数为零的点。trim函数从一起始点开始搜索，使用一个顺序二次程序设计算法，直到找到最近的trim点。用户必须隐含或明示地提供起始点。如果trim函数不能找到trim点，则返回搜索时遇见的状态微分在最小-最大意义上的最接近零的点，也就是，返回使微分对零的最大偏移为最小值时的点。trim函数可以发现满足具体的输入、输出或状态条件的trim点。

格式一：命令[x,u,y] = trim('sys')可以找到最接近系统初始状态x0的平衡点。特别的，trim可以找到使[x x0,u,y]的最大绝对值最小的平衡点。如果trim不能找到接近系统初始状态的平衡点，则返回系统接近平衡的点。特别的，它返回使abs(dx - 0)最小的点。用户可以使用下面的命令获得x0：

```
[sizes,x0,xstr] = sys([ ],[ ],[ ],0)
```

格式二：[x,u,y] = trim('sys',x0,u0,y0)找到接近x0,u0,y0的trim点，即，使abs([x —x0;u—u0;y—y0])最大值最小的点。

格式三：trim('sys', x0, u0, y0, ix, iu, iy)可以找到满足特定状态、输入和/或输出条件的最接近于x0,u0,y0的trim点。整数矢量ix，iu和iy必须在x0,u0和y0中选择合适的值。如果trim不能找到能精确满足指定条件的平衡点，则返回满足条件的最近点，即

abs([x(ix) - x0(ix);u(iu) - u0(iu);y(iy) - y0(iy)])

格式四：[x,u,y,dx] = trim('sys', x0, u0, y0, ix, iu, iy, dx0, idx)寻找特定的非平衡点，也就是，系统的状态微分有某些指定的非零值。这里，dx0指定搜索起点的状态微分值，idx在dx0中选择搜索精确满足的值。

可选项参数是一个由trim传递到最优化函数的优化参数数组。最优化函数是用来搜索

trim 点的。接着，最优化函数使用这个数组来控制优化过程，并返回关于该过程信息。trim 在搜索过程结束时返回数组内容。通过这种方式揭示优化过程，trim 允许用户监视和微调要搜索的 trim 点。

五个优化数组元素在寻找 trim 点时是非常有用的。表 5.4 描述了每个元素对搜索 trim 点的影响。

表 5.4 元素取值及说明

序号	默认值	描 述
1	0	指定显示内容。0 指定不显示；1 指定列表输出； 1 禁止警告信息
2	0.000 1	结束搜索 trim 点必须获得的计算精度
3	0.000 1	trim 搜索目标函数结束必须获得的计算精度
4	0.000 1	结束搜索状态微分必须获得的计算精度
10	N/A	无影响返回，是用来寻找 trim 点的迭代数

欲知详情，可查看最优化工具箱(Optimization Toolbox)有关说明。

5.10.4 应用举例

考虑线性状态空间模型

A，***B***，***C***，和 ***D*** 矩阵在名为 sys 的系统中，值如下所示：

$$\boldsymbol{A} = [-0.09 \quad -0.01; \quad 1 \quad 0];$$
$$\boldsymbol{B} = [\ 0 \quad -7; \quad 0 \quad 2];$$
$$\boldsymbol{C} = [\ 0 \quad 2; \quad 1 \quad 5];$$
$$\boldsymbol{D} = [\ 3 \quad 0; \quad 1 \quad 0];$$

例 1：寻找平衡点，使用命令

```
>> [x,u,y,dx,options] = trim('sys')
x =
     0
     0
u =
     0
y =
     0
     0
dx =
     0
     0
```

采取的迭代数为

```
>>option(10)
ans =
```

```
     7
```

例2:寻找接近 x = [1;1], u = [1;1]的平衡点,输入命令

```
>>x0 = [1;1];
>>u0 = [1;1];
[x,u,y,dx,options] = trim('sys', x0, u0);
x =
    1.0e-11
    -0.1167
    -0.1167
u =
    0.3333
    0.0000
y =
    -1.0000
    0.3333
dx =
    1.0e-11
    0.4214
    0.0003
```

采用迭代数为

```
>>option(10)
ans =
    25
```

例3:寻找输出固定为1的平衡点,使用命令:

```
>>y = [1;1];
>>iy = [1;2];
>> [x,u,y,dx] = trim('sys', [], [], y, [], [], iy)
x =
    0.0009
    -0.3075
u =
    -0.5383
    0.0004
y =
    1.0000
    1.0000
dx =
    1.0e-16
    -0.0173
```

```
    0.2396
```

例4:寻找输出固定为1且导数设置为0和1的平衡点,使用

```
>>y = [ 1;1 ];
>>iy = [ 1;2 ];
>>dx = [ 0;1 ];
>>idx = [ 1;2 ];
>> [x,u,y,dx,options] = trim('sys',[ ],[ ],y,[ ],[ ],iy,dx,idx)
x =
    0.9752
   -0.0827
u =
   -0.3884
   -0.0124
y =
    1.0000
    1.0000
dx =
    0.0000
    1.0000
```

采用迭代数为

```
>>option(10)
ans =
    13
```

5.10.5 限制条件

trim从任何给定的起始点开始搜索到的trim点只是一个局部值,其他更适当的trim点有可能存在。因此,如果想要为一个特别的应用程序找到最适当的trim点,应该尝试更多个x,u和y起始值。

5.10.6 命令算法

trim使用一个顺序二次程序设计算法寻找trim点,算法的具体内容详见"最优化工具箱"。

第6章 Simulink 高级仿真模型创建方法

对于初学者而言，掌握了前几章的仿真模型编辑、动态调试、仿真运行和结果分析等 Simulink 基本方法，就具备了完成一般性任务的基本条件。但对于需要创建更大、更复杂、应用面更宽和更优化的 Simulink 仿真模型的读者，还需要进一步学习和掌握高级模型的创建方法。

主要内容：Simulink 高级模块定制和子系统创建的封装编辑技术，Simulink 仿真原理及创建复杂系统应注意的事项。

学习目的：学会高级模块和模型的创建方法。

6.1 关于模块定制和子系统的封装技术

以前我们涉及的，包括将在第 7 章罗列的绝大多数模块，实际上均为系统的库模块，虽然涵盖面(包括其他应用工具内的模块)已经相当广泛，但对于一个具体应用而言，特别是很复杂和多层模型，不仅需要 Simulink 库模块，还需要用户自己定义一些模块集或子系统。这样复杂的系统模型，在仿真前就必须对在不同层次内的模块设置许多参数，这显然是不方便的。幸好，Simulink 为我们提供了便利的解决手段。封装技术是 Simulink 的一大特色，可以帮助用户为子系统定制对话框和图标，使模型层次清晰，调试方便，运行可靠。Simulink 提供的封装技术，可完成以下任务：

- 将子系统中的多个模块对话框简化为一个对话框。用户不需要打开每个模块的对话框来设置参数值，这些参数值可以在一个封装对话框中输入，并自动传递给封装子系统内的模块。
- 提供良好的用户界面。用户可以在对话框中定义自己模块的描述、参数栏标签，以及帮助文本。
- 定义计算变量的命令。该变量的值依赖于模块参数。
- 创建描述子系统用途的图标。
- 将子系统的内容隐藏于用户定制的界面之后，以防不经意间修改。
- 创建动态对话框。

编者提示：对于任何子系统都可以利用 Simulink 封装技术创建子系统的参数对话框、图标、描述、帮助及动态对话框等。

6.2 一个封装子系统的示例

图 6.1 为一个简单模拟直线方程 $y = mx + b$ 的子系统。

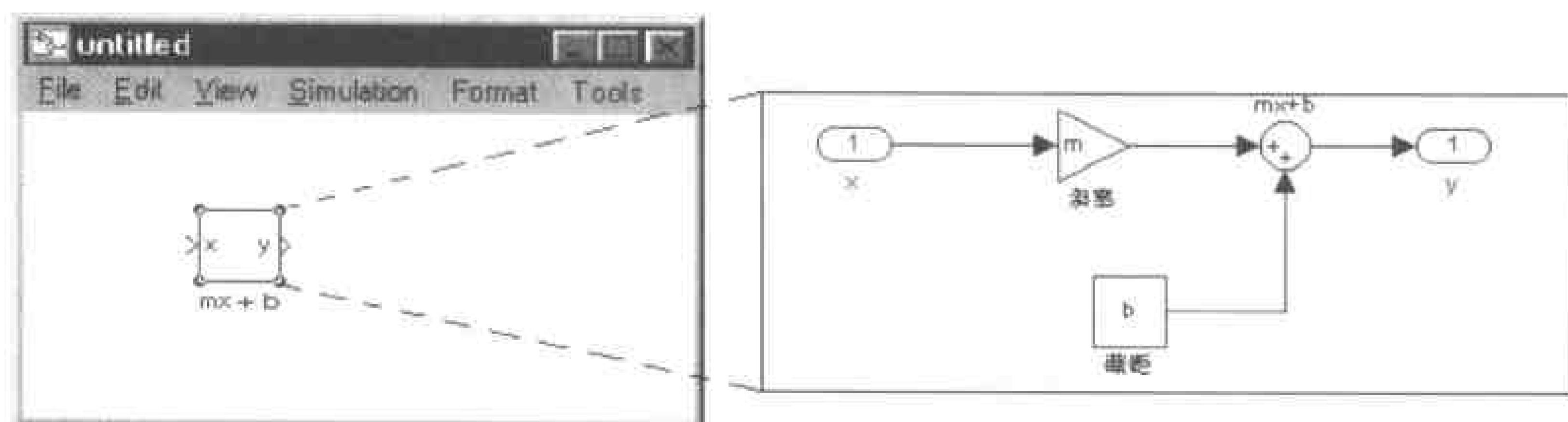


图 6.1 子系统范例

子系统 mx+b 包含一个名为斜率的 Gain(增益)模块,其增益参数设置为 m;一个名为截距的 Constant(常数)模块,其常量值参数为 b,这两个参数分别表示直线的斜率和截距。

本例中为该子系统创建一个自定义对话框和图标。对话框包含斜率和截距提示。在封装之后,双击该 mx+b 模块打开封装对话框。封装对话框和图标如图 6.2 所示。

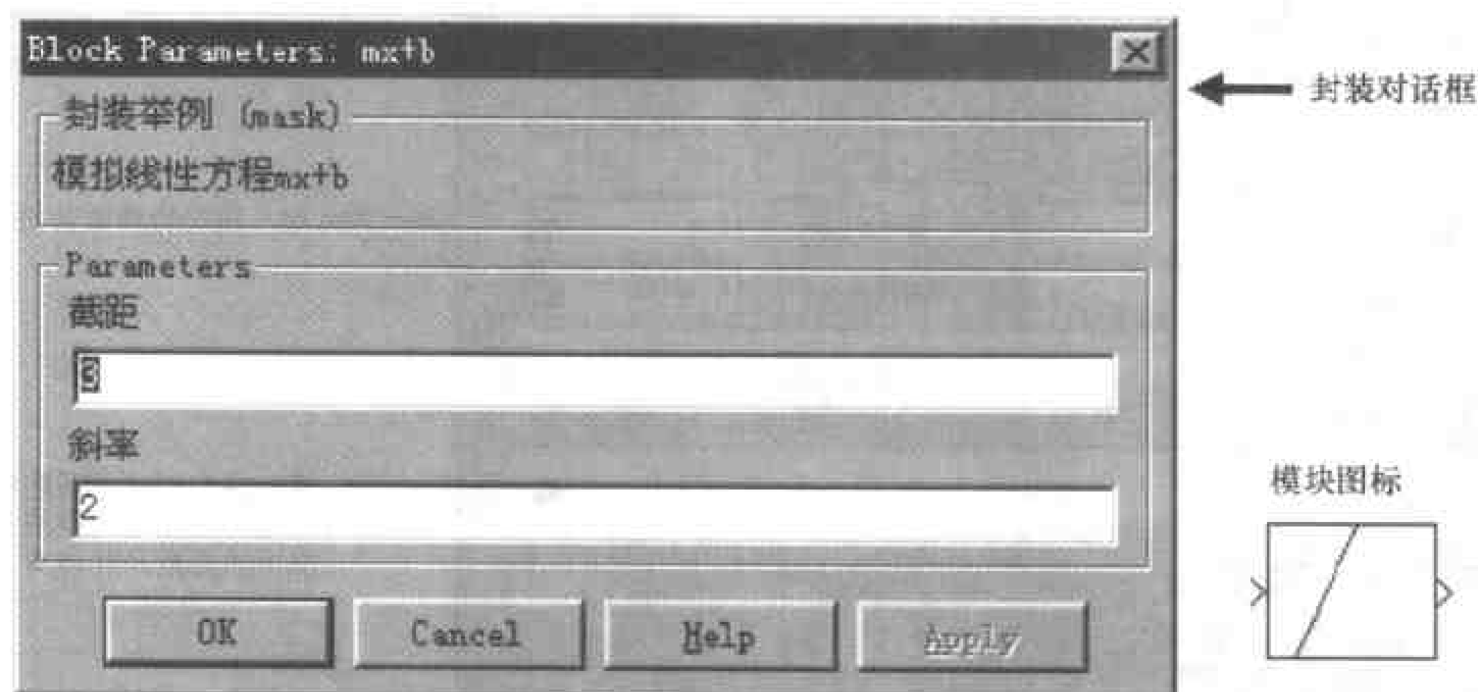


图 6.2 封装对话框及图标

用户在封装对话框中的参数栏分别输入截距和斜率值,Simulink 将这些值应用于所有子系统内层的模块。封装本子系统也就是创建了一个自我包含的功能单元(功能模块),它有自己的指定应用参数:截距和斜率。封装的过程将这些被封装的参数映射到内层模块的参数。子系统的复杂性被封闭在一个看起来就如同与 Simulink 内置模块一样的新的界面中。

要对本子系统进行封装,用户需要做下面几项工作:

- 指定封装对话框参数的提示。在本例中,封装对话框有斜率和截距两个提示。
- 指定用来存储各个参数值的变量名。
- 输入模块文本记录,包括模块描述及模块帮助文本。
- 指定创建模块图标的制图命令。
- 指定提供制图命令所需要的变量命令(本例中不涉及)。

封装子系统是靠 Simulink 提供的封装编辑器,在后面各节介绍封装编辑器的使用方法时,读者可以比较本范例。

在封装之前，先按照图 6.1 中右图的结构建立子系统(编者提示：m 变量作为"增益"模块的值；b 变量作为"常数"模块的值)。建立子系统的方法有多种，在第 2,3 章已经做过介绍，如需要，请参阅 3.10 一节。

6.2.1　创建封装对话框中的提示及相关信息

为了封装子系统，选中 Subsystem 模块并选择 Edit 菜单中的 Mask Subsystem 项。本节开始时显示的封装对话框大部分在封装编辑器(Mask Editor)的 Initialization 页创建。在这个例子中，对话框提示及说明如图 6.3 所示。

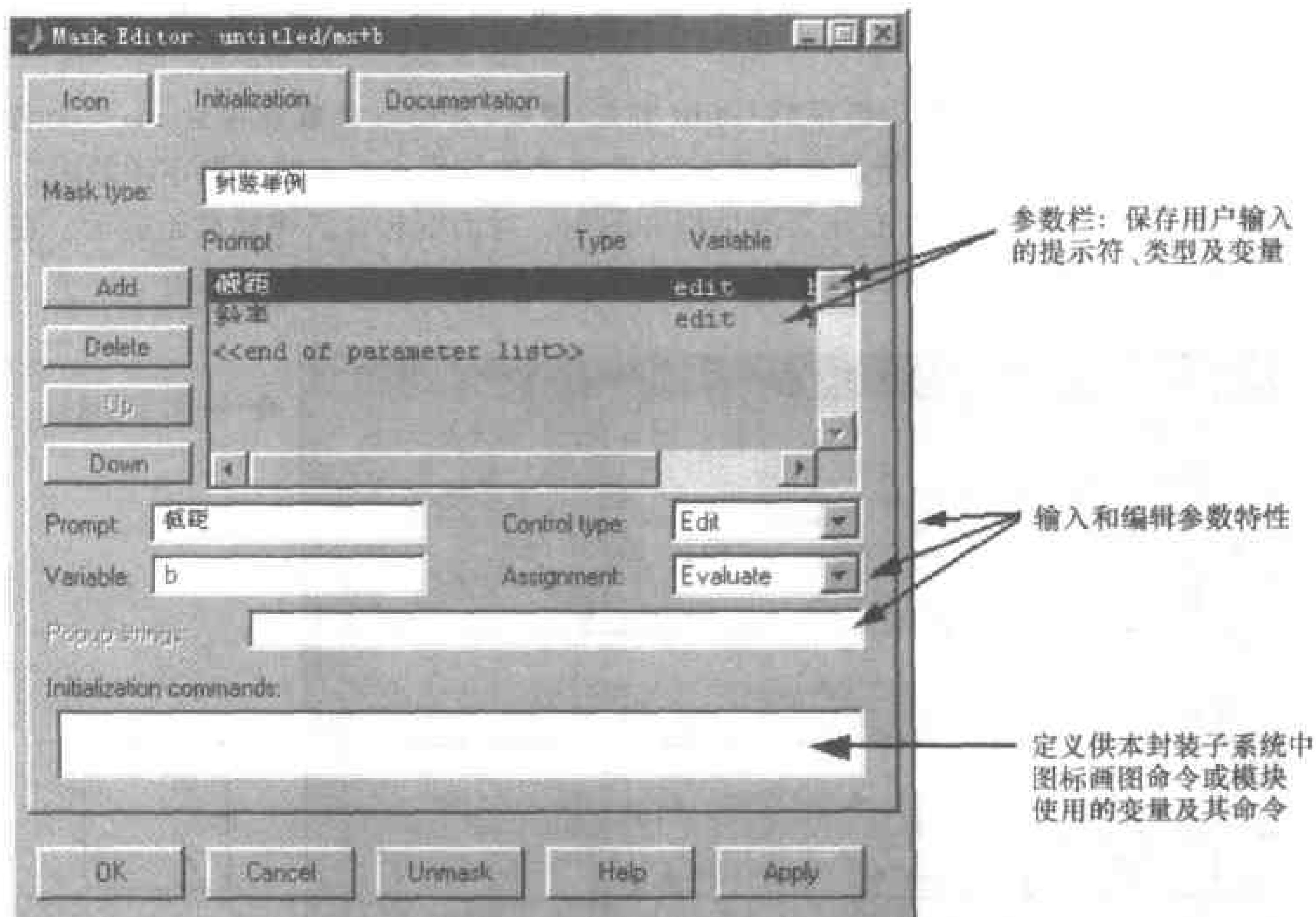


图 6.3　提示信息和说明

封装编辑器(Mask Editor)让用户指定封装参数的如下属性：

- 提示：描述参数的文本标签。
- 控制类型：用户界面控制的类型，确定参数值是输入还是选择。
- 变量：要存储参数值的变量名。

通常情况下，通过提示符来指导封装参数。在本例中，与斜率对应的参数提示符为斜率，与截距对应的参数提示符为截距。

斜率和截距均被定义为编辑控件(edit)，这表示用户要在封装对话框的编辑栏中键入参数值。这些值储存在 mask workspace(封装工作空间)内的变量中(详见"初始化命令"一节中的"封装工作空间")。封装模块只能访问封装工作空间的变量。在这个例子中，输入的斜率值赋予变量 m。封装子系统中的斜率模块，从封装工作空间获得斜率参数值作为斜率参数。图 6.4 显示了封装编辑器中的截距和斜率参数定义与映射到实际的封装对话框的参数。

在为斜率和截距创建了封装参数之后，按 OK 按钮。然后，双击 mx+b 子模块，打开新建

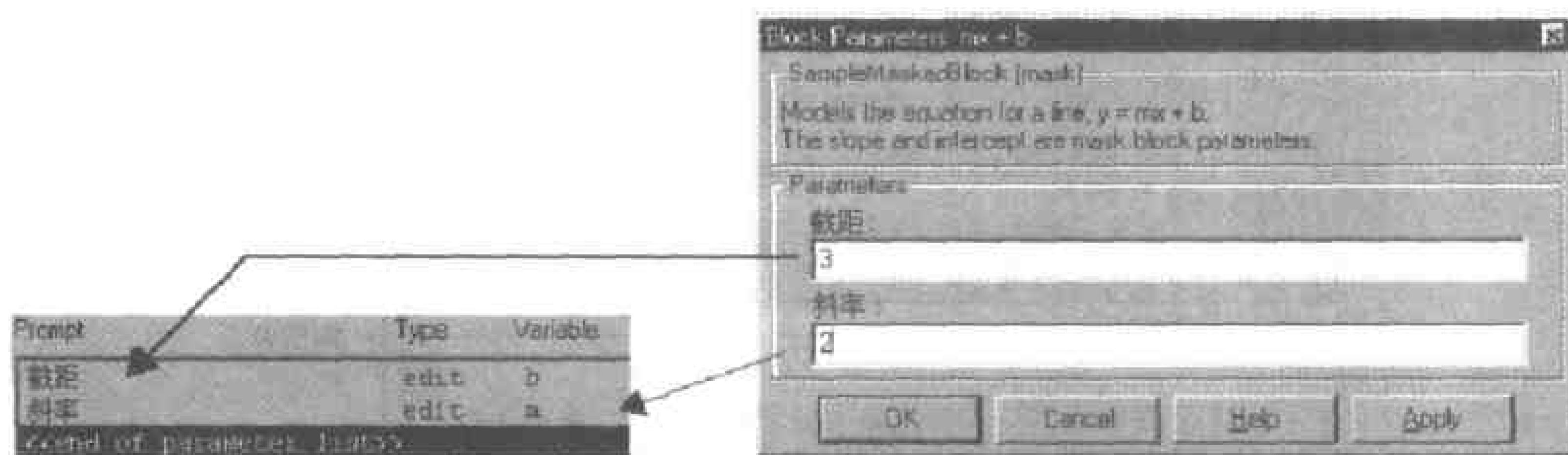


图 6.4 参数的映射关系

的参数对话框，给截距参数项输入 3，斜率参数项输入 2。

6.2.2 创建模块描述和帮助文本

封装类型、模块描述及帮助文本都在 Documentation 选项中定义。对本例的封装模块，其 Documentation 如图 6.5 所示。

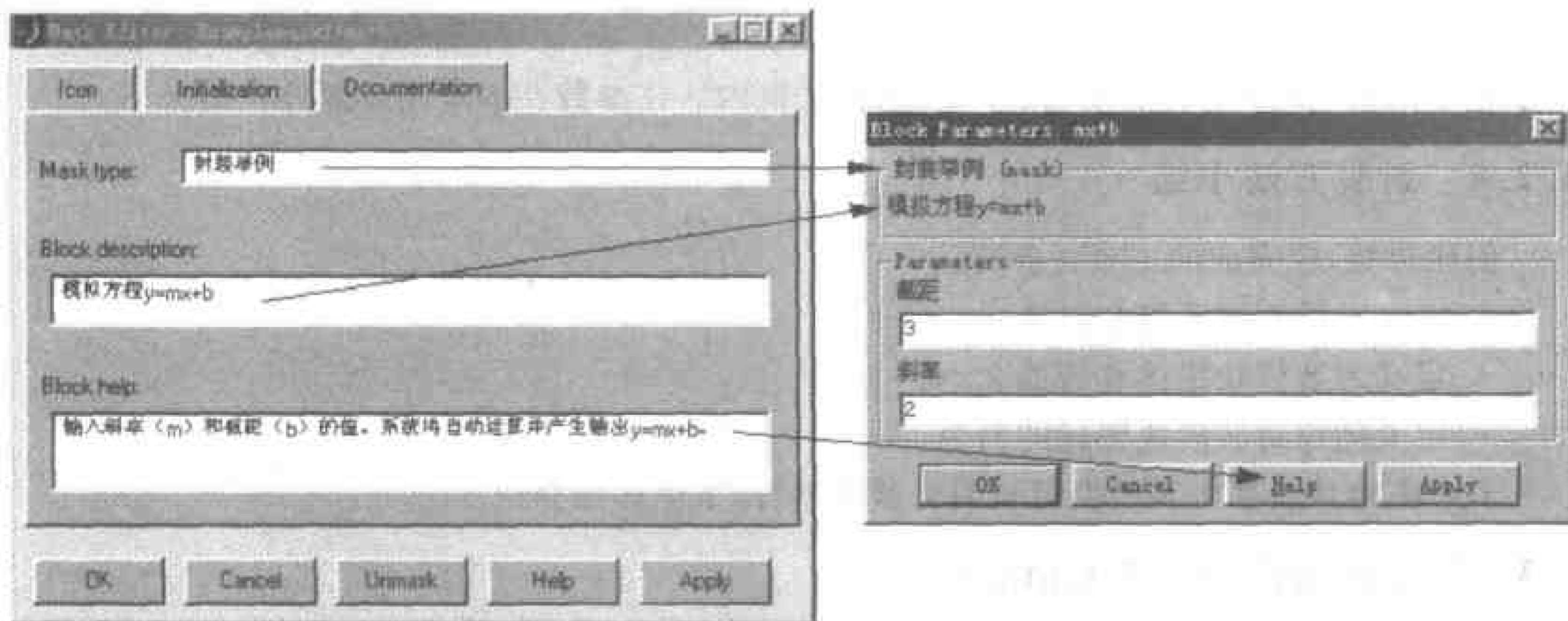


图 6.5 封装对话框相关显示与定义

6.2.3 创建模块图标

至此，我们已经为直线 mx + b 子系统创建了自定义对话框，然而，这个子系统模块显示的仍然是一般 Simulink 子系统的图标，本例中封装模块的图标是指示直线斜率的图标。如斜率为 3 时，其图标如图 6.2 所示。

模块图标的定义在 Icon 选项，对于本例的模块，图标选项如图 6.6 所示。

制图命令绘制一条从(0,0)到(0,m)的直线。如果斜率是负的，Simulink 将起点抬高到 1 以使该线在模块的可见范围内。

制图命令能访问封装工作空间所有的变量。如果输入不同的斜率值，图标就会更新所画线的斜率。

选择位于图标属性列表底部的 Drawing coordinates 参数为 Normalized，指定的图标是在

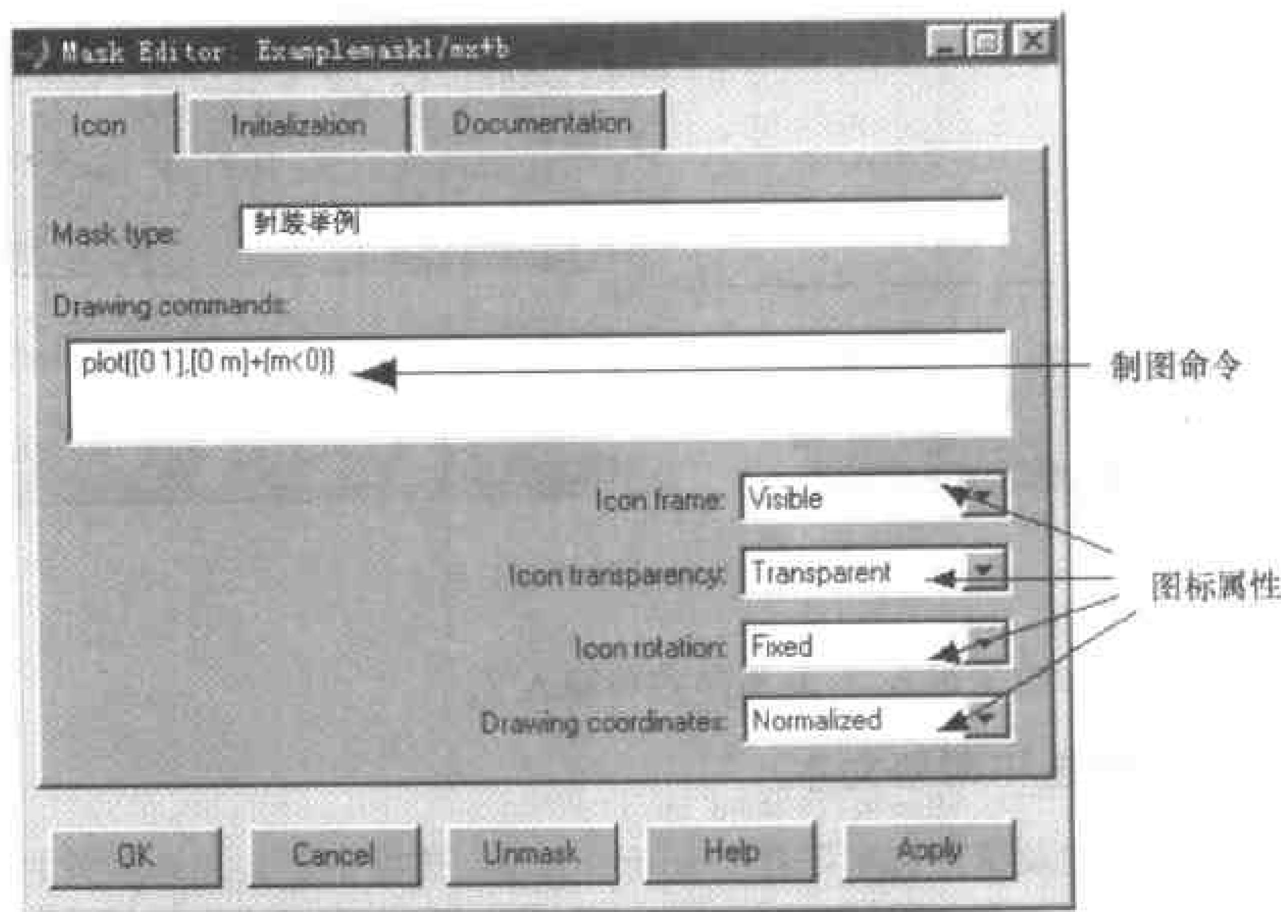


图 6.6 图标定义及说明

一个左下角为(0,0),右上角为(1,1)的格式框内。该参数在本章稍后部分介绍。

6.2.4 封装方法小结

创建封装,应完成的主要任务有:

- 定义对话框提示符和特性;
- 定义封装模块描述和帮助文本;
- 定义创建封装模块图标的命令。

子系统的封装都需要借助封装编辑器来进行,下面详细介绍。

6.3 封装编辑器(Mask Editor)

请先确定要封装的是一个 Subsystem 模块(或是一个通过 3.10 节所述的创建的子系统)。选中 Subsystem 模块,然后选中 Edit 菜单上的 Mask Subsystem(如果不是 Subsystem 模块或子系统,该选项是不可选的),弹出封装编辑器,如图 6.7 所示,它由三个选项组成,分别进行不同的封装任务:

- Initialization 选项。用于定义和描述封装对话框的参数提示,定义与参数关联的变量名,以及指定初始化命令。
- Icon 选项。用于定义模块图标。
- Documentation 选项。用于定义封装类型,指定模块描述和模块帮助。

封装编辑器底部有五个按钮:

- OK 按钮:应用所有选项的封装设定,并关闭封装编辑器。
- Cancel 按钮:关闭封装编辑器,但并不应用在上一次按 Apply 按钮后所做的设定和修改。
- Unmask 按钮:退出封装并关闭封装编辑器,但封装信息是保存的,以便为再次激活封

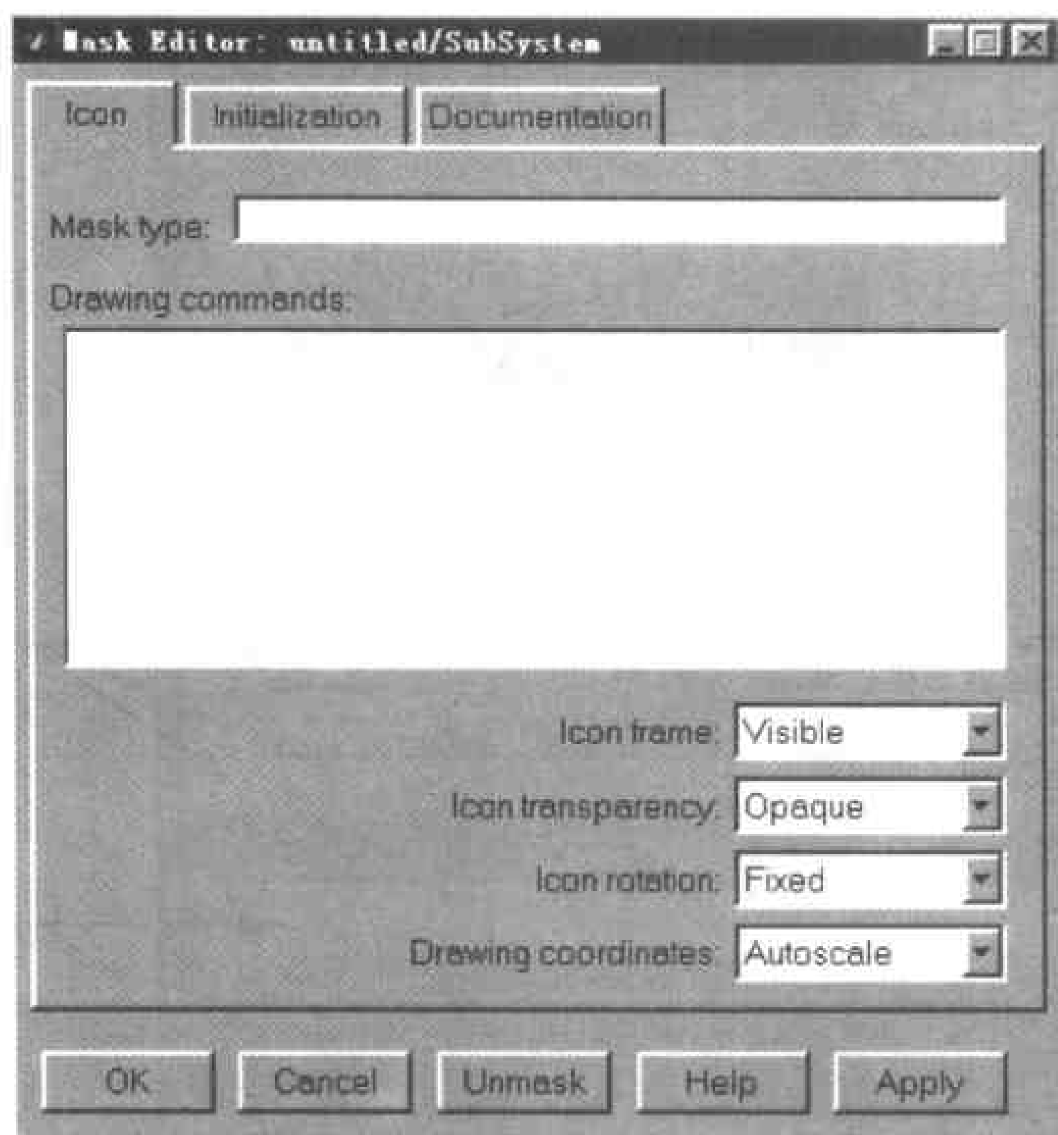


图 6.7 封装编辑器

装使用。要再次激活封装，选中模块并选择 Create Mask。封装编辑器打开，且显示以前的设置。只有当模型关闭并不能恢复时，无效封装信息才被放弃。

- Help 按钮：显示帮助(英文)。
- Apply 按钮创建封装，或应用所有封装选项所做改变和设定。封装编辑器仍然打开。

要不解除封装查看封装之下的系统内容，先选中该 Subsystem 模块，从 Edit 菜单中选择 Look Under Mask。这个命令打开该子系统，且模块的封装不受影响。

6.3.1 初始化(Initialization)选项

通过封装界面，用户可以为封装系统内部的模块输入参数值。这项任务可以通过在 Initialization 选项定义参数值提示来创建封装界面。对于范例 mx+b 系统，定义 Initialization 选项如图 6.8 所示。

1. 提示符及相关变量

编辑提示符的目的是向用户提供模块参数值的输入和选择信息，提示符将以在 Prompt 列表上的次序出现在封装后的对话框上。

当定义一个提示符时，同时也必须指定存储该参数值的变量，选择该提示符的控制类型，并指示参数值如何储存在变量中。

如果 Assignment(赋值)类型是 Evaluate，用户输入的值在派给该变量之前是由 MATLAB 计算的。如果类型是 Literal，则用户输入的值不由计算产生，而是以字符串的格式派给该变量。

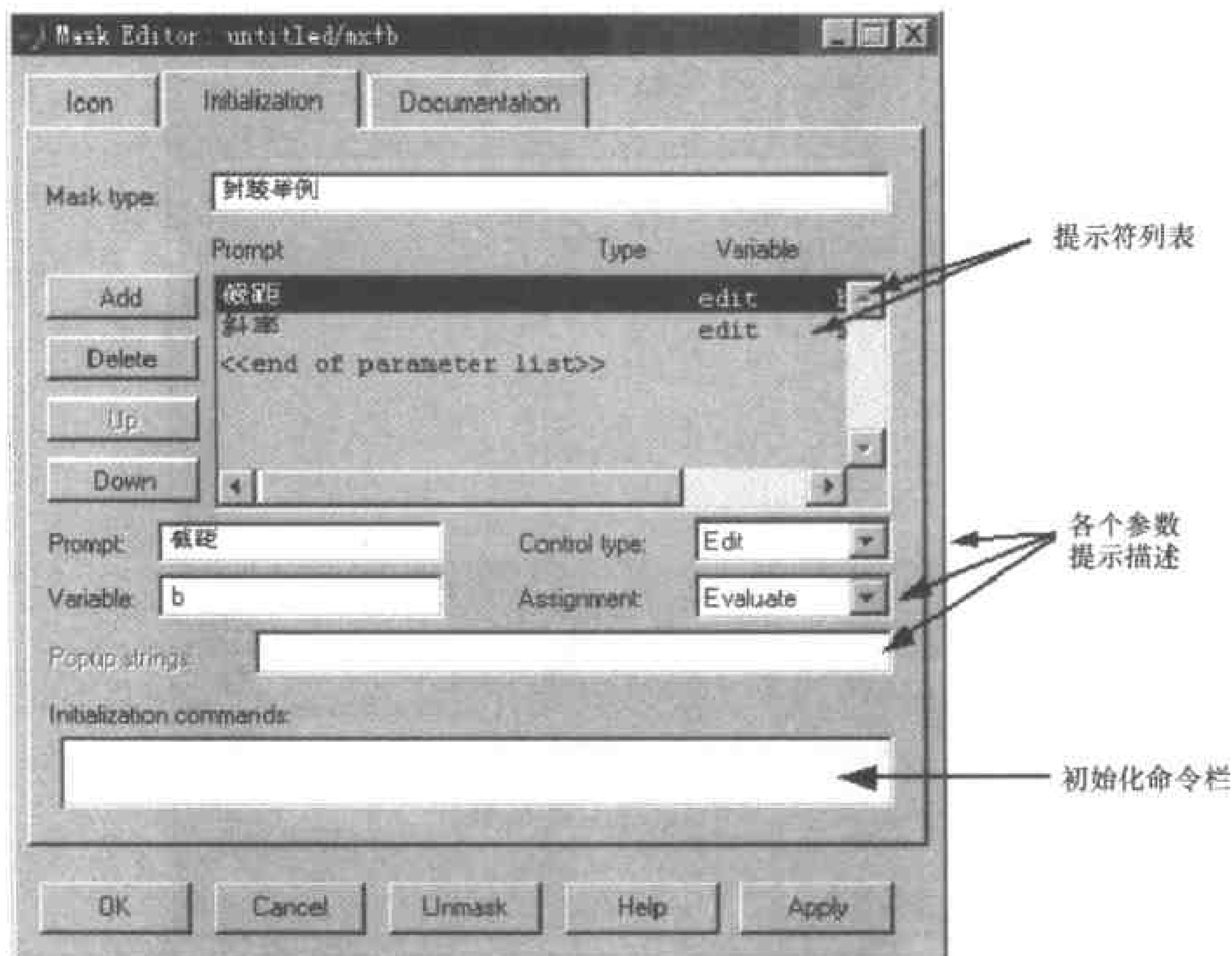


图 6.8　定义模块封装界面及说明

例如，如果用户在编辑域内输入字符串 gain，且 Assignment 类型是 Evaluate，则字符串 gain 是由 MATLAB 计算，其结果赋值给变量。如果类型是 Literal，则字符串不由 MATLAB 计算，即变量包含字符串‘gain’。

如果用户需要输入的参数值既有字符串，又有需要计算的值，则选择 Literal，然后在初始化命令栏中输入 MATLAB 的 eval 命令。举例来说，如果 LitVal 变量是字符串‘gain’，那么要获得计算值，使用命令：

```
value = eval(LitVal)
```

一般情况下，大多数参数是 Evaluate 的赋值(Assignment)类型。

(1) 创建第一个提示符。在列表中创建第一个提示符时，在 Prompt 栏内单击，然后输入提示符(如：斜率)；在 Variable 栏内单击(此时，“斜率”等提示符自动列表排序)，输入储存该参数值的变量(如：m)，并选择一种控制类型和赋值类型(如：edit)。

编者提示：如果 Prompt 不可选，先单击 Add 按钮。

(2) 插入提示符。在列表中插入一个提示符：

① 选中插入点的提示符，即插入的提示符将出现在该提示符之前。单击提示符列表左边的 Add 按钮。

② 在 Prompt 域内输入提示文字，在 Variable 域内输入保存该参数值所对应的变量。

(3) 编辑提示符。编辑已经存在的提示符：

① 在列表中选中该提示符。提示符、变量名、控制类型及赋值类型就会出现在列表的下

面。

② 编辑适当的值。在区域外单击鼠标或按 Enter 键或 Return 键，旧的提示符就会自动被更新。

(4) 删除一个提示符。想要从列表中删除一个提示符：

① 选中要删除的这个提示符。

② 单击提示符列表左边的 Delete 按钮来删除。

(5) 移动提示符。在列表中移动提示符：

① 选中要移动的提示符。

② 单击提示符列表左边的 Up 按钮可以将提示符在列表中向上移动一个位置；欲向下移动一个位置，单击 Down 按钮。

2. 控制类型(Control type)

Simulink 提供了三种参数值输入或选择控制方式：编辑(Edit)，复选框(Checkbox)和下拉菜单(Popup)控制。例如，图 6.9 显示了一个同时使用三种参数域控制方式的封装对话框(包括弹出式控制)范例。

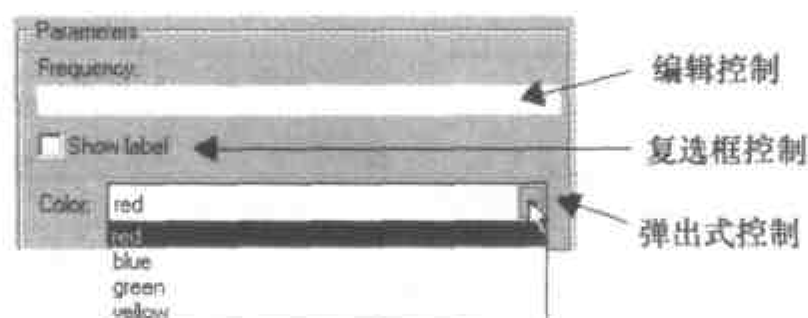


图 6.9 参数控制对话框

(1) 定义编辑控制。用户可以通过在一个区域内输入参数值来设置参数。图 6.10 和表 6.1 描述了图 6.9 所示例子的提示符编辑控制的定义过程及含义。

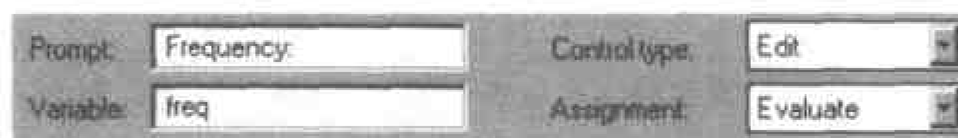


图 6.10 变量(Variable)栏

与参数(freq)关联的变量值是由提示符定义的 Assignment 类型决定的。

表 6.1 变量类型及参数值

Assignment 类型	参数值(Value)
Evaluate	对在编辑栏内输入的表达式的计算结果
Literal	在编辑栏内输入的字符串

(2) 定义复选框控制。选择复选框控制，可以让用户通过复选框在两个选项之间选择一项。图 6.11 及表 6.2 显示了复选框控制的定义过程。

与参数(label)关联的变量值取决于复选框是否被选中，且提示符的 Assignment 类型是

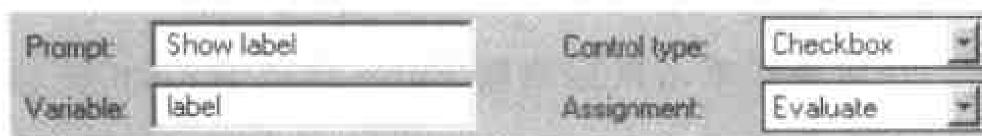


图 6.11 复选框及含义

否被定义。

表 6.2 复选框及状态值

复选框	Evaluated 值	Literal 值
选中	1	'on'
未选中	0	'off'

(3) 定义弹出式列表控制。弹出式控制设置,可以让用户从列表中选择参数值。用户可以在 Popup strings 栏内指定列表,用垂直线"|"分隔每一项。图 6.12 显示了弹出式控制是如何定义的。

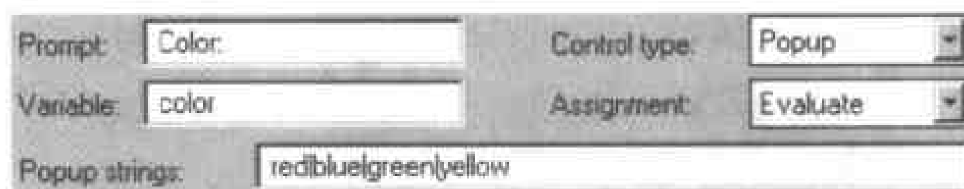


图 6.12 弹出式列表控制

与参数(color)关联的变量值取决于从弹出式列表中选择的项,及提示符的 Assignment 类型。

表 6.3 选项及参数

Assignment 类型	参数值
Evaluate	为所选择的项在列表中的序号,从 1 开始。例如,列表的第三项被选中,则参数值为 3
Literal	所选中的字符串。例如,第三项被选中,参数值为"green"

3. 编辑封装模块参数的默认值

要改变封装的库模块的默认参数值,需要进行下列步骤:

- 解锁含有本模块的函数库。
- 打开模块进入它的对话框,填写想要的默认值,关闭对话框。
- 保存该库。

当该模块拷贝到一个模型中和打开后,默认值出现在模块对话框中。

关于库的更多信息,参见第 3 章。

4. 设置可调参数

可调参数是一种用户在运行期间可以改变值的封装参数。当用户创建一个封装时,它的

所有参数都是可调的。通过 MaskTunableValues 参数，可以随后设置任何模块的参数为不可调或再次可调。这个参数的值是字符串单元数组，每个数组单元对应一个封装模块的参数。第一个单元对应第一个参数，第二单元对应第二个参数，依次类推。如果参数是可调的，相应的单元值为'on'；否则为'off'。要使一个参数可调或不可调，必须使用 MATLAB 命令，先用 get_param 命令得到其对应的数组单元，然后设置相应的数组单元为'on'或'off'，并使用 set_param命令重新设置 MaskTunableValues 参数的值。例如，下面的命令使当前所选的封装模块的第一个参数为不可调：

```
>>ca = get_param(gcb,'MaskTunableValues');
>>ca(1) = 'off'
>>set_param(gcb,'MaskTunableValues', ca)
```

在改变一个模块的可调参数后，保存该模块即保存了所做的变化。

5. 初始化命令(Initialization Commands)

初始化命令用来定义位于封装工作空间的变量。这些变量能被所有用于定义封装的初始化命令，所有封装子系统内的模块，以及所有用于绘制模块图标的命令(绘图命令)共同使用。

Simulink 执行初始化命令的时机：

- 模型装载。
- 仿真开始运行，或者模块结构经过更新。
- 封装模块已旋转。
- 模块图标需要重新绘制且绘图命令依赖于在初始化命令中定义的变量。

初始化命令由有效的 MATLAB 表达式组成，包括 MATLAB 函数、运算符和在封装工作空间定义的变量。初始化命令不能访问基本工作空间内的变量。应在每句初始化命令之后加一个分号，以避免在命令窗口立即响应结果。

(1) 封装工作空间。在下列情况下，Simulink 创建一个局部工作空间，叫做封装工作空间(mask workspace)：

- 封装包含初始化命令。
- 封装定义了提示符和与这些提示符关联的变量。

封装的模块不能访问基本工作空间或其他的封装工作空间。封装工作空间的内容包括与封装参数对应的变量和由初始化命令定义的变量。封装工作空间中的变量能被封装模块访问。如果模块是一个子系统，这些变量还可以被所有子系统内部的模块访问。

封装工作空间与 M 文件函数使用的局部工作空间类似。可以将输入到内部模块对话框的表达式和输入到 Mask Editor 的初始化命令看作 M 文件函数的程序行。使用这样的类比，局部工作空间可以看作是该封装工作空间"函数"。

在本章前面讨论的例子 mx+b 中，Mask Editor 通过将变量与封装参数关联制定了 m 和 b。然而，封装工作空间中的变量并不是被显式地指定到封装内部的模块，而是封装内部的模块能访问该封装工作空间的所有变量。这样，可以将封装内部的模块形象地看作是"向内观察"封装工作空间。

图 6.13 显示了从封装对话框输入的值与封装工作空间的变量的映射关系(由实线指示)，以及内部模块对这些变量的访问关系(由虚线指示)。

(2) 调试初始化命令。可以通过下面的方法调试初始化命令：

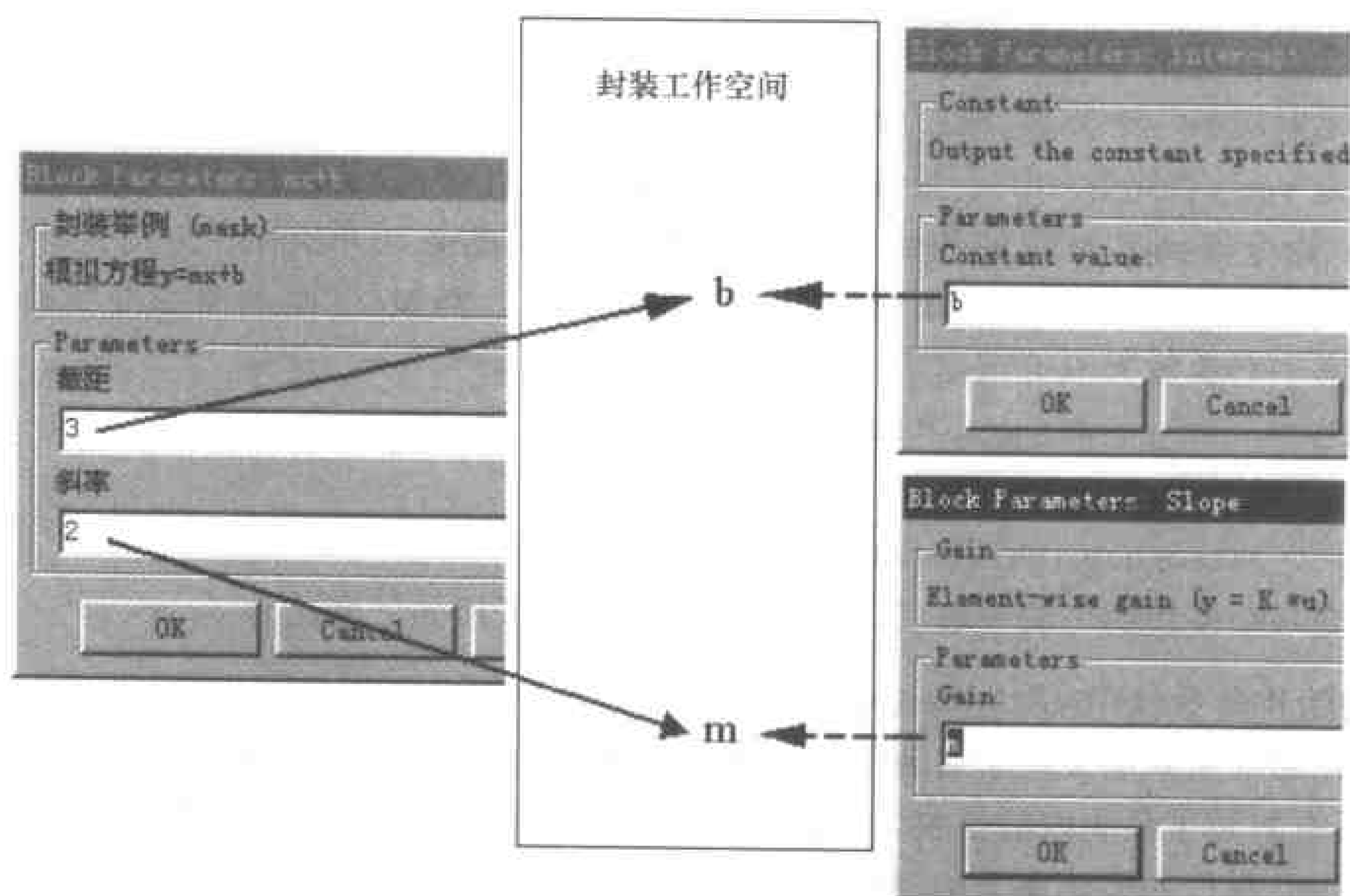


图 6.13　输入值与工作空间变量、模块与变量的关系

- 指定初始化命令末尾不加分号，在命令窗口上显示响应结果。
- 在初始化命令中加入键盘(keyboard)命令来停止执行和把控制权交给键盘。更多信息，请参见 keyboard 命令的帮助文本。
- 在 MATLAB 命令窗口输入下列命令中的任何一个：

```
dbstop if error
dbstop if warning
```

如果在初始化命令中有错误发生，执行停止，用户可以检查封装工作空间。更多信息，参见 dbstop 命令的帮助文本。

6.3.2　Icon 选项

Icon 选项用于定制封装模块的图标。通过在 Drawing commands 栏内指定命令，可以定制用户化的图标。图标可以是描述文本、状态方程式、图像、图形等。Icon 选项对话框如图 6.14所示。

绘图命令可以访问该封装工作空间的所有变量。

绘图命令可以显示文字、一个或多个图，或者显示一个传递函数。如果用户输入多个命令，其结果依命令的次序绘制在图标上。

1. 在图标中显示文字

(1) 命令格式。要在图标中显示文字，输入下列绘图命令之一：

```
disp('text') or disp(variablename)
text(x, y, 'text')
text(x, y, stringvariablename)
```

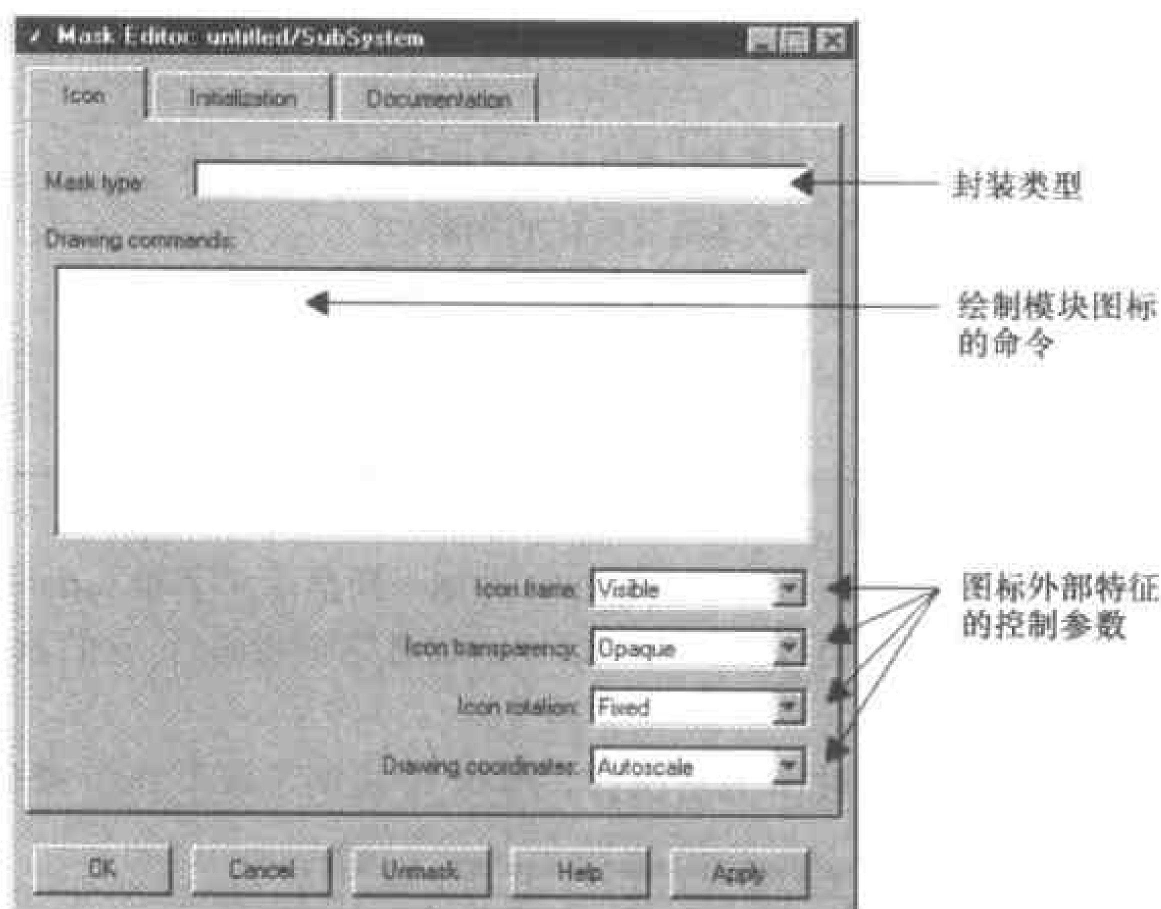


图 6.14 Icon 对话框

```
text(x, y, text, 'horizontalAlignment', halign,'verticalAlignment', valign)
fprintf('text') or fprintf('format', variablename)
port_label(port_type, port_number, label)
```

(2) 命令描述：

① disp 命令在图标中心显示 text 或者 variablename 的内容。

② text 命令在指定点(x,y)的地方放置一字符串(text 或是 stringvariablename 的内容)。其单位取决于 Drawing coordinates 参数。更多信息，参看本节后部的“控制图标属性”。

在 text 命令中，用户可以选择指定文本相对于点(x,y)水平和/或垂直对齐(见表 6.4、6.5)。例如，命令：

```
text(0.5, 0.5, 'foobar', 'horizontalAlignment', 'center')
```

使 foobar 在图标中居中显示。

表 6.4 text 命令提供的水平对齐选项

选项	对齐方式
left	文本的左边限对齐于指定点
right	文本的右边限对齐于指定点
center	文本的水平中心对齐于指定点

表 6.5　text 命令提供下面的垂直对齐选项：

选项	对齐方式
Base	文本的基线对齐于指定点
bottom	文本的底线对齐于指定点
middle	文本的垂直中心对齐于指定点
cap	文本的大写字母线对齐于指定点
top	文本的顶端线对齐于指定点

③ Fprintf 命令显示居于图标中心的格式文本，并能一同显示文字和 variablename 的内容。

编者提示：当这些命令名与相应的 MATLAB 函数名完全相同时，它们只提供上面描述的功能。

要显示超过一行的文字，使用\n 表示换行。图 6.15 表明了两个 disp 命令的不同应用。

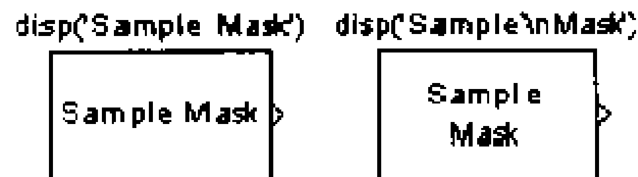


图 6.15　disp 命令示例

④ port_label 命令指定在图标上显示端口的标签。该命令的句法如下：

```
port_label(port_type, port_number, label)
```

其中 port_type(端口类型)是 'input'或 'output'，port_number(对应于端口类型的端口数)是一整数，label 是指定该端口标签的字符串。例如命令：

```
port_label('input', 1, 'a')
```

指定"a"为输入端口 1 的标签。

2. 在模块图标中显示图形

除了在封装模块的图标中显示文本，还可以通过输入绘图命令显示图形。可以使用下面的命令形式：

```
plot(Y);
plot(X1,Y1,X2,Y2,...);
```

如果 Y 是矢量，plot(Y)按照矢量元素的排序依次绘制相应的点。如果 Y 是一个矩阵，则将矩阵的每一列看作矢量元素绘制。

plot(X1,Y1,X2,Y2,...)画出矢量 Y1 对 X1 的曲线，Y2 对 X2 的曲线，等等。矢量对的长度必须相同，并且该列表必须由偶数个矢量组成。



图 6.16　Ramp 模块图标

例如，下面的命令在 Sources 库中的 Ramp 模块图标上绘制图形(图 6.16)：

```
plot([0 1 5], [0 0 4])
```

绘图命令可以包含 NaN 和 inf 值。当遇到 NaN 或 inf 时，Simulink 停止绘图，在下一个非 NaN 或 inf 的数开始重画。

图标上的图形的效果还取决于 Drawing coordinates 参数的值。更多信息，参看“控制图标属性”。

有时候，在模块图标上会显示三个警告问号(？？？)，这说明有以下情形发生：

- 当绘图命令中使用的参数还没有定义时(例如，当封装第一次创建，且参数值还没有输入到封装对话框中时)。
- 当一个封装模块参数或绘图命令输入不正确时。

3. 在图标上显示图像

使用封装对话函数 image 和 patch 可以在封装模块图标上显示位图和填充颜色。image(a) 显示图像 a，其中 a 是一个 RGB 值的 $M \times N \times 3$ 的数组。还可以使用 MATLAB 命令 imread 和 ind2rgb，读取位图文件和将其转换为矩阵格式。举例来说，命令

```
image(imread('icon.tif'))
```

从 MATLAB 当前路径下的一个名为 icon.tif 的 TIFF 文件中读取图标图像。

(1) image 命令：

命令 image(a, [x, y, w, h])在相对于封装模块左下角的指定点位置创建图像。

命令 image(a, [x, y, w, h], rotation)可以指定在图标旋转时，该图像是否随之旋转('on')，或者保持不动('off')。默认值为'off'。

(2) patch 命令：

patch(x, y)创建一个由坐标矢量 x 和 y 勾画的实体颜色图像。填充颜色是当前的前景色。

patch(x, y, [r g b])创建一个实体颜色图像，其颜色由向量[r g b]指定，其中 r 是红色分量，g 是绿色，b 为蓝色。例如：

```
patch([ 0 .5 1], [0 1 0], [1 0 0])
```

为在封装图标上显示一个红色的实三角形。

4. 在模块图标上显示传递函数

如果想在模块图标上显示一个传递函数方程式，须在 Drawing commands 栏内输入下列命令：

```
dpoly(num, den)
dpoly(num, den, 'character')
```

其中 num 和 den 是传递函数分子和分母系数矢量，由初始化命令定义。方程式以 character 指定的字母显示，默认为 s。当图标画好后，初始化命令执行，结果方程式显示在图标上。

- 要以 s 的降幂显示一个连续的传递函数，输入

```
dpoly(num, den)
```

如果 num = [0 0 1], den = [1 2 1]，图标显示如图 6.17 所示。

$$\frac{1}{s^2+2s+1}$$

图 6.17 图标显示的传递函数

- 以 z 的降幂显示一离散的传递函数，输入

  dpoly(num, den, 'z')

如果 num = [0 0 1],den = [1 2 1],图标显示如图 6.18 所示。

$\frac{1}{z^2+2z+1}$

图 6.18 图标显示的传递函数

- 以 1/z 的升幂显示一离散的传递函数，输入

dpoly(num, den, 'z-')

如果 num 和 den 定义如前，图标显示如图 6.19 所示。

$\frac{z^2}{1+2z^{-1}+z^{-2}}$

图6.19 图标显示的传递函数

- 显示零-极点增益传递函数，输入

  droots(z, p, k)

如，z = []; p = [-1 -1]; k = 1,图标显示如图 6.20 所示。

$\frac{1}{(s+1)(s+1)}$

图 6.20 图标显示的传递函数

还可以再加一个表达式('z' or 'z-'),使方程式以 z 或者 1/z 的形式表示。

在创建图标时，如果参数没有定义或没有设置值，Simulink 将在图标上显示三问号(???)。当参数值输入到封装对话框，Simulink 计算传递函数并在图标显示结果方程式。

5. 控制图标属性

通过在 Drawing commands 栏的下面各选项，可以控制封装模块图标的属性。

(1) Icon frame　Icon frame 指的是图标外的矩形边框。设置 Icon frame 参数为 Visible，显示边框；若设置为 Invisible，隐藏该边框。默认值为显示图标边框。例如，图 6.21 显示了 AND(与)门模块的图标边框可见和不可见的区别。

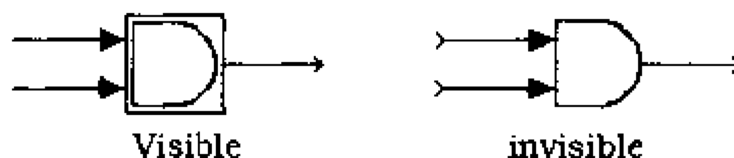


图 6.21 图标边框设置的效果

(2) Icon transparency　Icon transparency 指的是图标的透明度。图标可以设置为 Opaque(白底)或者 Transparent(透明)，即隐藏或显示图标下的内容。默认为白底，覆盖 Simulink 图画信息，如端口标签等。图 6.22 显示了 AND(与)门模块的白底和透明图标的区别。注意透明图标上的文本。

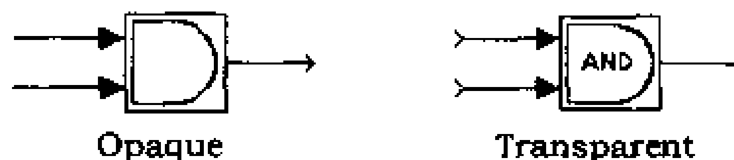


图 6.22 图标透明的效果

(3) Icon rotation　Icon rotation 指的是图标旋转。当模块旋转和翻转时，可以选择图标是否随之旋转和翻转，或者保持在原来的方向不动。其默认是不旋转图标，或图标旋转与模块端口旋转一致。图 6.23 显示的是，当 AND(与)门模块旋转时，选择 Fixed 和 Rotates 项的不同效果。

(4) Drawing coordinates　该参数控制绘图命令使用的坐标系统，只适用于 plot 和 text



图 6.23 图标的旋转效果

绘图命令。可选项有：Autoscale，Normalized 和 Pixel（图 6.24）。

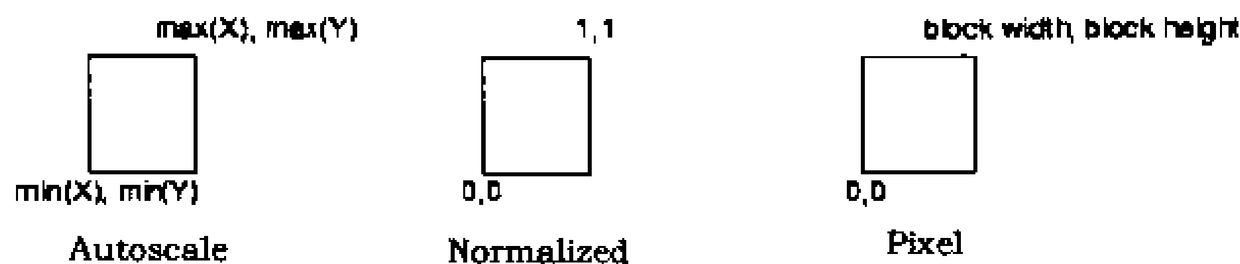


图 6.24 绘图选项

• Autoscale：在模块边框内自动调整图标的大小。当模块是可调整大小时，其图标也是可调整大小的（图 6.25）。例如，使用下列矢量显示图标。

$$X = [\ 0\ 2\ 3\ 4\ 9\ ];\ Y = [\ 4\ 6\ 3\ 5\ 8\ ];$$

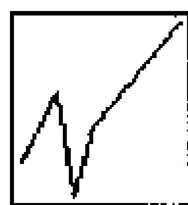


图 6.25 定义调整模块尺寸



图 6.26 定义调整模块尺寸

模块的左下角边框坐标是(0,3)，右上角坐标是(9,8)。X 轴的范围是 9(从 0 到 9)，而 Y 轴的范围是 5(从 3 到 8)。

• Normalized：图标在模块边框内，其左下角坐标(0,0)，右上角坐标为(1,1)。只有 X 和 Y 值在 0 和 1 之间才能显示出来。当模块是可调整大小时，其图标也是可调整大小的(图 6.26)。例如，使用下列矢量显示图标：

$$X = [.0\ .2\ .3\ .4\ .9];Y = [.4\ .6\ .3\ .5\ .8];$$

• Pixel：X 和 Y 的值以像素为单位表示的图标。当模块是可调整大小时，图标并不自动地调整大小。要迫使图标随着模块调整大小，必须依据模块大小定义绘图命令。

下面的例子演示了如何为前面讨论的例子 mx+b 封装子系统创建一个改进的图标。不论模块的形状大小，这些初始化命令都可以使绘图命令产生精确图标：

```
pos = get_param(gcb, 'Position');
width = pos(3) - pos(1); height = pos(4) - pos(2);
x = [0, width];
if (m >= 0), y = [0, (m * width)]; end
if (m < 0), y = [height, (height + (m * width))]; end
```

而产生图标的绘图命令是 plot(x,y)。

6.3.3 Documentation 选项

Documentation 可以定义或改变封装模块的类型、描述文本和帮助文本。图 6.27 表示了 mx+b 例子的 Documentation 选项下的编辑框和封装模块的对话框之间的对应关系。

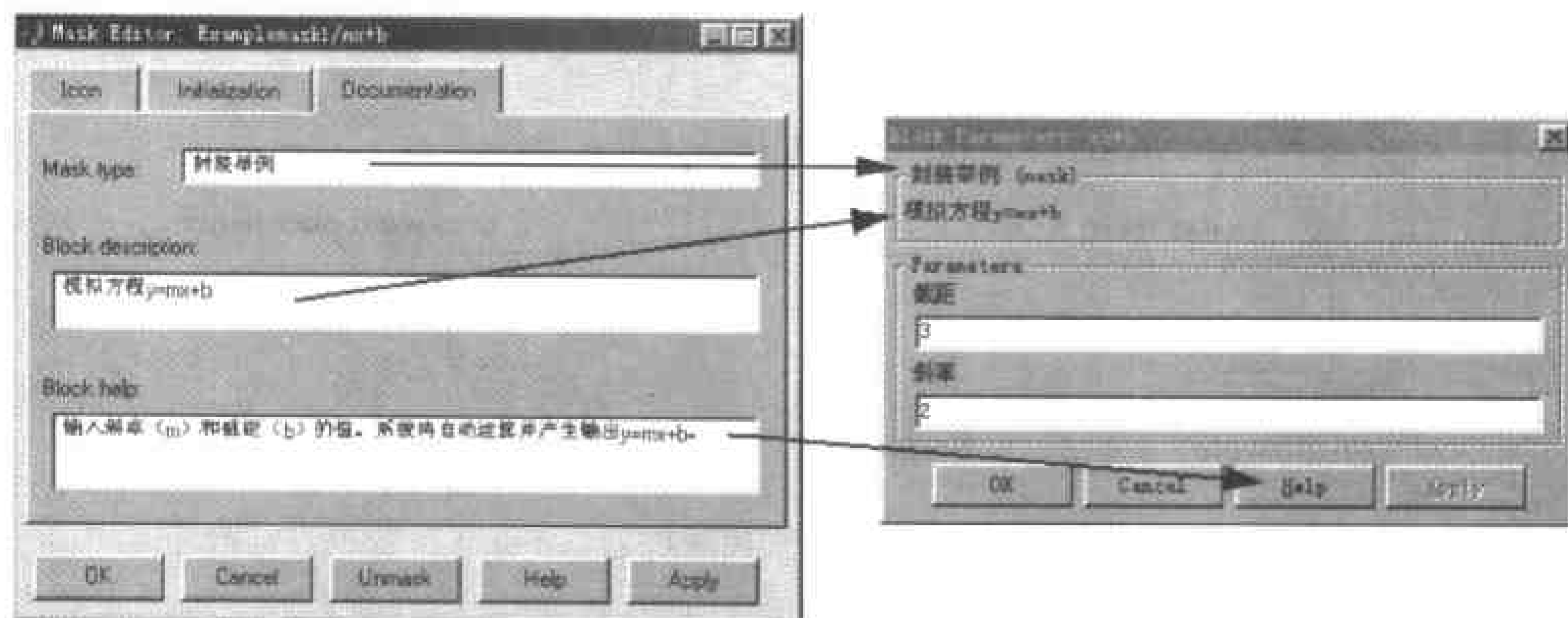


图 6.27 编辑框和封装模块的对话框之间的对应关系

1. Mask Type 编辑框

Mask Type(封装类型)仅仅是用于文件管理目的的模块分类,它出现在模块对话框和所有模块的 Mask Editor 上。用户可以为封装模块类型选择任何名称。当 Simulink 创建该模块的对话框时,在封装类型之后加上“(mask)”来区分封装模块和内置模块。

2. Block Description 编辑框

Block Description(模块描述)是出现在模块对话框的 Mask Type 之下的信息文本。如果是为其他用户设计一个系统,在此输入对模块的目的和功能的描述是大有益处的。Simulink 对任意长度的文本进行自动换行,也可以使用 Enter 或 Return 键换行。

3. Block Help 编辑框

可以编辑帮助文本,在封装模块对话框中的 Help 按钮按下时,能显示该帮助文本。如果是为他人使用而创建模型,这是解释模块如何工作和如何设置参数的好地方。

封装模块帮助可以包含用户手写文本。可以为封装模块帮助文本指定下列任何文本的类型:

- URL 规约(以 http:, www, file:, ftp:或 maiilo:开始的字符串);
- web 命令(启动一个浏览器);
- eval 命令(计算一个 MATLAB 字符串);
- Web 浏览器的静态文本显示。

Simulink 检测封装模块帮助文本的首行。如果它检测为 URL 规约,web 命令或 eval 命令,它按照指导访问模块帮助。否则,封装模块帮助文本的内容将全部显示在浏览器中。

下面列出了一些可接受的命令。

```
web([ docroot '/My Blockset Doc/' get_param(gcb,'MaskType') '.html'])
```

```
eval('! Word My_Spec.doc')
http://www.mathworks.com
file:///c:/mydir/helpdoc.html
www.mathworks.com
```

Simulink 对过长的文本进行自动换行处理。

6.3.4　为封装模块创建动态对话

Simulink 提供创建封装模块的动态人机对话的功能，在用户输入发生变化时，其外部特性也随之自动改变。封装对话的可变特征包括下面几种变化：

• 参数控制的立现性。比如改变一个参数能控制另一参数的出现或消失。当一个控制出现或消失时，对话相应展开或收缩。

• 参数控制的使能状态。改变一个参数可起到控制另一参数输入的使能或禁止。Simulink 自动将禁止的控制设置为灰色，在视觉上指示其禁止了。

• 参数值。改变一个参数能将相关的参数设置为需要的值。

创建动态封装对话需要同时使用 mask editor（封装编辑器）和 Simulink 的 set_param 命令。首先，使用封装编辑器定义所有的对话参数，包括静态和动态的。接着，在 MATLAB 命令行输入 set_param 命令，指定对话响应用户输入的调回函数。最后，保存包含该封装子系统的模型或库，就完成了封装模块的动态对话的创建。

1. 设置封装模块对话参数

Simulink 定义了一组封装模块参数，用来指定当前封装模块对话的状态。可以使用封装编辑器来检查和制定其中的参数。get_param 和 set_param 等 Simulink 命令，也可以用来检查和制定封装对话参数。使用这些命令的好处是，set_param 命令允许用户制定参数，并因此在该对话打开的时候改变对话外部特性，接下来就可以创建动态的封装对话。

例如，可以在 MATLAB 命令行输入 set_param 命令，当用户改变一个用户定义的参数时，指定调回函数的调用。该调回函数接着可以使用 set_param 命令来改变先前定义（预定义）的封装对话参数的值和状态（如隐藏或显示，使能或禁止一个用户自定义的参数控制）。

2. 预定义封装对话参数

Simulink 把下面的预定义参数与封装对话联系起来。

(1) MaskCallbacks　这个参数的值是一个字符串单元数组，为对话框中自定义参数控制指定调回表达式。第一个单元定义第一个参数控制的调回，第二单元为第二个参数定义控制，依次类推。调回可以是任何有效的 MATLAB 表达式，包括激活 M 文件的命令表达式。这就表明，可以用 M 文件实现复杂调回任务。

封装对话最容易的调回设置方法是，首先在一个模型里或库窗口中选中相应的封装对话，然后在 MATLAB 命令行输入 set_param 命令。例如，如下命令

```
>>set_param(gcb,'MaskCallbacks',{'parm1_callback', '', 'parm3_callback'});
```

为当前选中的模块的封装对话定义了第一个加第三参数调回。要保存该调回设定，应保存包含该封装模块的模型或库。

(2) MaskDescription　本参数的值是一指定模块描述字符串。设置本参数，可以动态地

改变模块描述。

(3) MaskEnables　本参数的值是一字符串单元数组，为本对话定义用户自定义参数控制的使能状态。第一个单元定义第一个参数控制的使能状态，第二个单元为第二参数定义控制的使能状态，等等。值'on'指示对用户输入相应的控制是使能(激活)的，值'off'指示该控制是禁止的。

可以通过在调回中设置此参数，动态地使能或禁止用户输入。例如，调回中的命令

```
>>set_param(gcb,'MaskEnables',{'on','on','off'});
```

将使当前打开的封装模块对话的第三控制无效。Simulink 将该控制变为灰色，表示禁止。

(4) MaskPrompts　本参数的值是一字符串单元数组，为用户自定义的参数指定提示符。第一个单元为第一个参数定义提示符，第二个单元为第二参数定义提示符，以此类推。

(5) MaskType　本参数的值是与当前对话相关联的封装模块的类型。

(6) MaskValues　本参数的值是一字符串单元数组，为对话指定自定义参数的值。第一个单元为第一个参数指定值，第二单元为第二参数指定值，依次类推。

(7) MaskVisibilities　本参数的值是一字符串数组单元，为对话指定自定义参数控制的可见性。第一个单元定义第一个参数的控制可见性，第二个单元为第二个参数定义，以此类推。值'on'表示相应的控制是可见的，值'off'表示该控制是隐藏的。

通过设置本参数，可以动态地隐藏或者显示用户自定义的参数控制。例如，调回中的命令

```
>>set_param(gcb,'MaskVisibilities',{'on','off','on'});
```

将隐藏当前选中的模块的第二个自定义封装参数的控制。Simulink 展开或者收缩对话来相应地显示或者隐藏控制。

6.4 条件执行子系统

条件执行子系统，指的是其执行取决于输入信号值的子系统。控制子系统执行的信号称为控制信号，该信号从 Subsystem 模块的控制输入端口输入。

在构建一个复杂模型时，有的模块的执行依赖于其他模块，此时，条件执行子系统就是非常有用的。

Simulink 支持三种类型的条件执行子系统：

• 使能子系统，在控制信号是正数时执行。它在控制信号时间步内，从负到正的方向穿过零值时开始执行，并在控制信号保持为正时继续执行。关于使能子系统的详细描述参看 6.4.1 节“使能子系统”。

• 触发子系统，在每次触发事件发生时执行一次。触发事件可以发生在触发信号的上升沿或是下降沿，触发信号可以是连续的或离散的信号。触发子系统的详细描述参看 6.4.2。

• 触发加使能子系统，在使能控制信号为正时，每次触发事件发生时执行一次。有关详细内容参看 6.4.3。

编者提示：Simulink 4.0 版本已将本节所述的各种子系统和其他新增加的子系统合成为一个 Subsystems 模块库，敬请注意。

6.4.1　使能子系统

使能子系统是在每个仿真步内的控制信号为正数时执行的子系统。使能子系统有一个控制输入口，输入可以是标量，也可以是矢量。

- 如果输入的控制信号是标量，则当输入值大于零时该子系统执行。
- 如果输入的控制信号是矢量，则当矢量的任何元素大于零时该子系统执行。

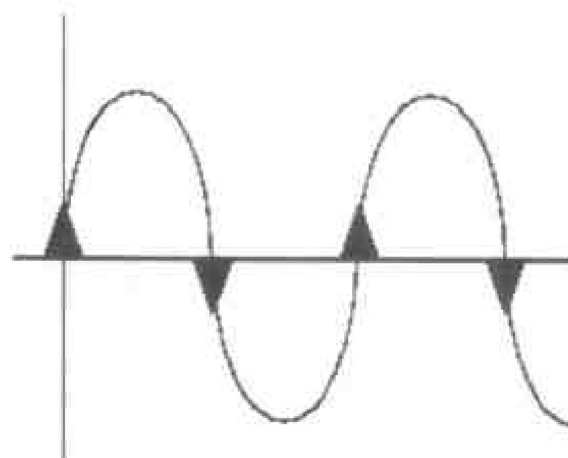


图 6.28　正弦波控制

例如，如果控制信号是一正弦波，则子系统交替地激活和禁止。如图 6.28 所示，上升箭头表示激活，下降箭头表示禁止。

Simulink 运用过零斜率的方法来确定使能事件是否发生。如果信号过零，并且斜率为正数，则子系统激活；如果在过零点斜率为负数，则子系统禁止。

1. 创建使能子系统

要创建使能子系统，可以从 Signals& Systems 库中拷贝一个 Enable(使能)模块到一个子系统，Simulink 系统自动给 Subsystem 模块的图标加一个使能符号和一个控制信号输入端口。如前所述，双击该模块，打开子系统模型，加入需要的模块和连线。

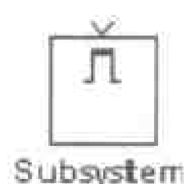


图 6.29　使能符号与控制信号端口

(1) 在子系统禁止时设置输出值。虽然当使能子系统禁止时，子系统并不执行，但仍然可以将输出信号提供给其他模块使用。在使能子系统禁止时，用户可以选择保持该子系统前一次执行后的输出值，或恢复初始状态。

打开各个 Outport 模块对话框，如图 6.30 所示，并为 Output when disabled 选项选中一个参数：

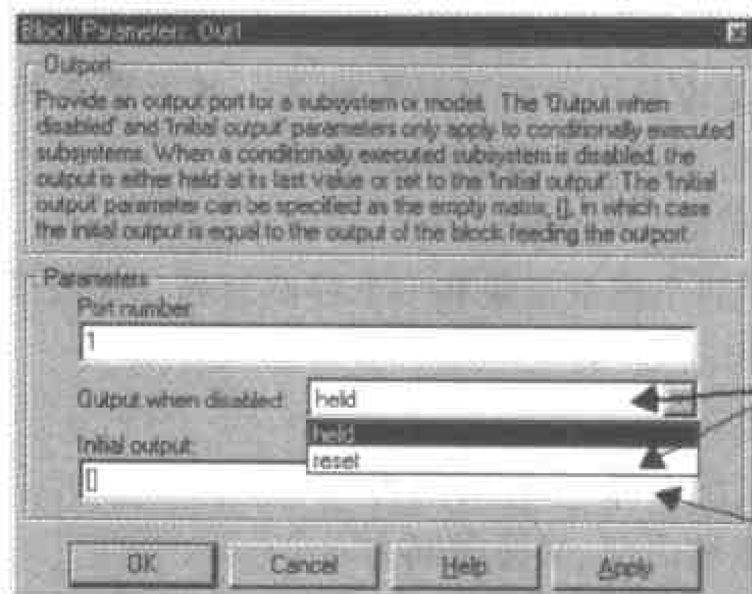


图 6.30　Outport 模块对话框及说明

- 选择 held，使输出保持最近的值。
- 选择 reset，使输出恢复到初始值。输出初始值在 Initial output 栏内设置。

(2) 设置子系统再次激活时的状态。当一个使能子系统执行时，用户可以选择是保持子

系统前一次执行后的状态值，还是复位到初始状态。

打开 Enable 模块对话框，如图 6.31 所示，并为 States when enabling 参数选择一个值：

- 选择 held 使状态保持为最近执行后的值。
- 选择 reset 使状态恢复到初始时的值。

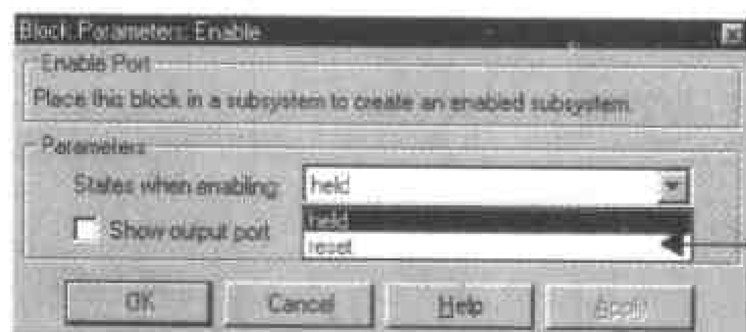


图 6.31 Enable 模块对话框

(3) 输出使能控制信号。Enable 模块对话框中的有一个 Show output port 复选框，用于是否输出使能控制信号的控制，如图 6.32 所示：

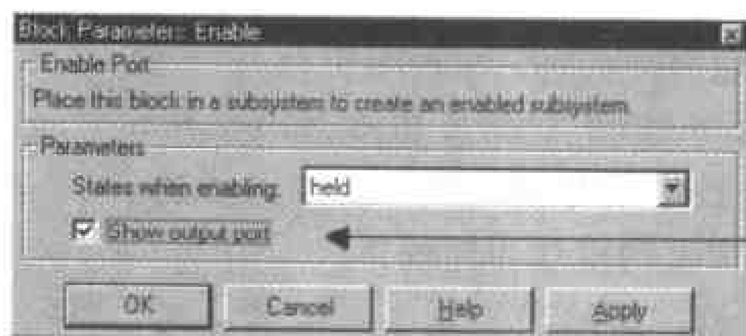


图 6.32 显示输出端口

选中复选框就表明允许把控制信号输出到使能子系统，这在将控制信号所包含的值作为使能子系统执行的判断逻辑这一点上是很有用的。

2. 使能子系统可以包含的模块

使能子系统可以包含任何模块，无论是连续的还是离散的。其中离散模块只有在子系统执行时，并且只有当它的采样时间与仿真采样时间同步时才执行。

编者提示：使能子系统与模型共用同一时钟。

例如，图 6.33 所示的系统包含了四个离散模块和一个控制信号。离散模块是：

- 模块 A(Block A)，采样时间为 0.25 s。
- 模块 B(Block B)，采样时间为 0.5 s。
- 模块 C(Block C)，在使能子系统内部，采样时间为 0.125 s。
- 模块 D(Block D)，在使能子系统内部，采样时间为 0.25 s。

使能控制信号由 Pulse Generator(脉冲发生器)模块产生，标签为 Signal E，它在 0.375 秒时由 0 变为 1，在 0.875 时返回到 0(图 6.34)。

模块 A 和 B 的执行与使能信号无关，因为它们不是使能子系统中的一部分。当使能信号变成正数时，模块 C 和 D 以它们各自指定的采样速率执行，直到使能信号再次变成零。

编者提示：在使能信号变为零时，模块 C 在 0.875 s 时并未执行。

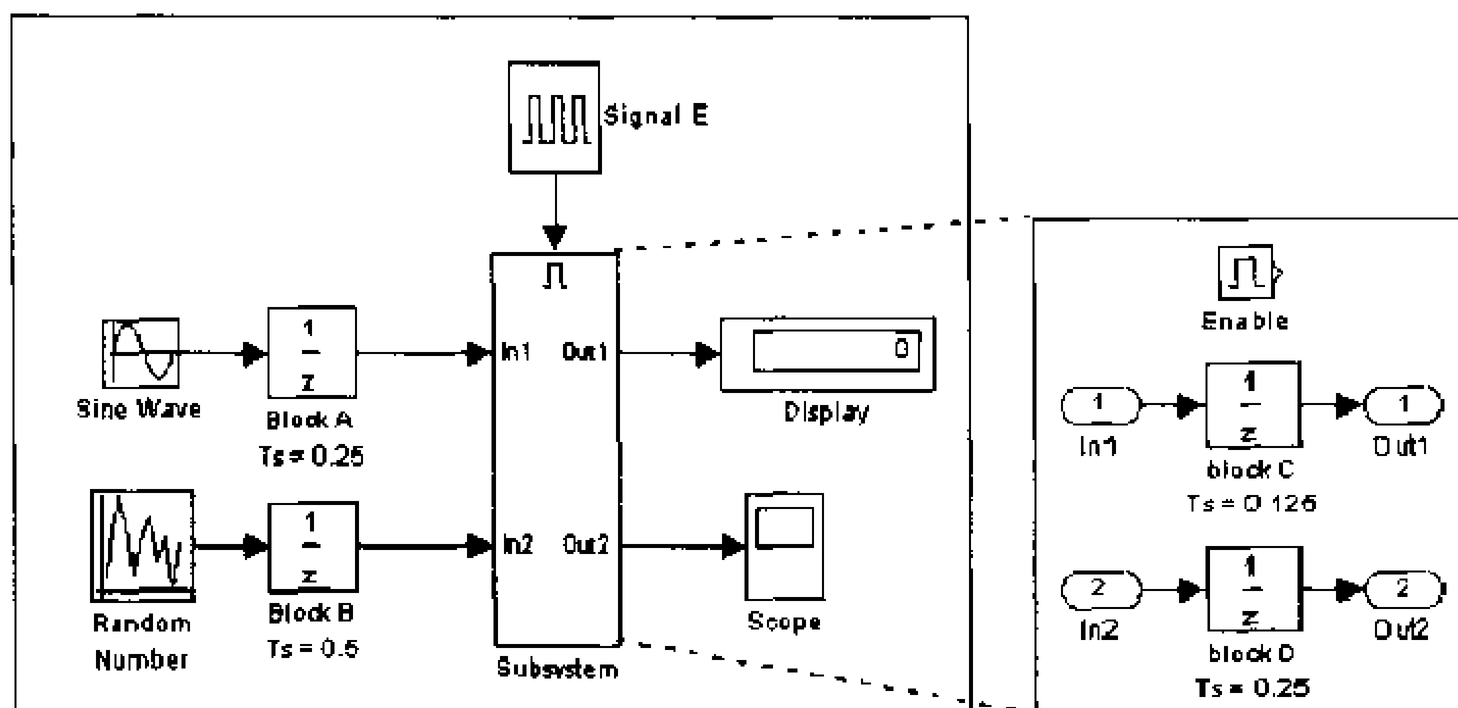


图 6.33 包含使能信号及端口的系统

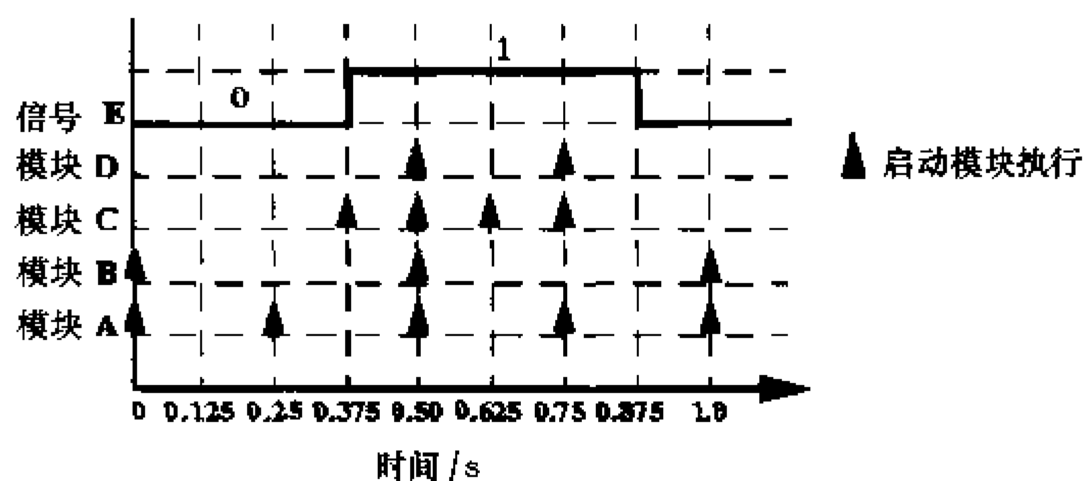


图 6.34 各个离散模块执行时序

6.4.2 触发子系统

触发子系统是在每次触发事件发生时执行的子系统。

一个触发子系统有一个控制输入端，叫做触发输入(trigger input)，它决定该子系统是否执行。可以从下面三种触发事件中选择一种作为条件，强制使触发子系统开始执行：

- 上升沿(rising)触发，当控制信号从负数或零上升到正数(或者当初始值为负数，上升到零)时，执行该子系统。
- 下降沿(falling)触发，当控制信号从正数或零下降为负数(或者当初始值为正数，下降到零)时，执行该子系统。
- 上升沿和下降沿同时(either)触发，无论上升沿还是下降沿都触发执行子系统。

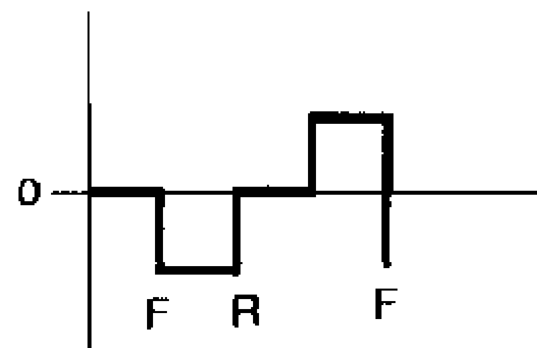


图 6.35 控制信号触发的时机

例如，图 6.35 标出了控制信号上升沿(R)触发和下降沿(F)触发的时机。

在如图 6.36 所示的例子中，子系统是在方波控制信号上升沿被触发的。

1. 创建触发子系统

创建触发子系统，可以从 Signals& Systems 库中拷贝一个 Trigger(触发)模块到一个子

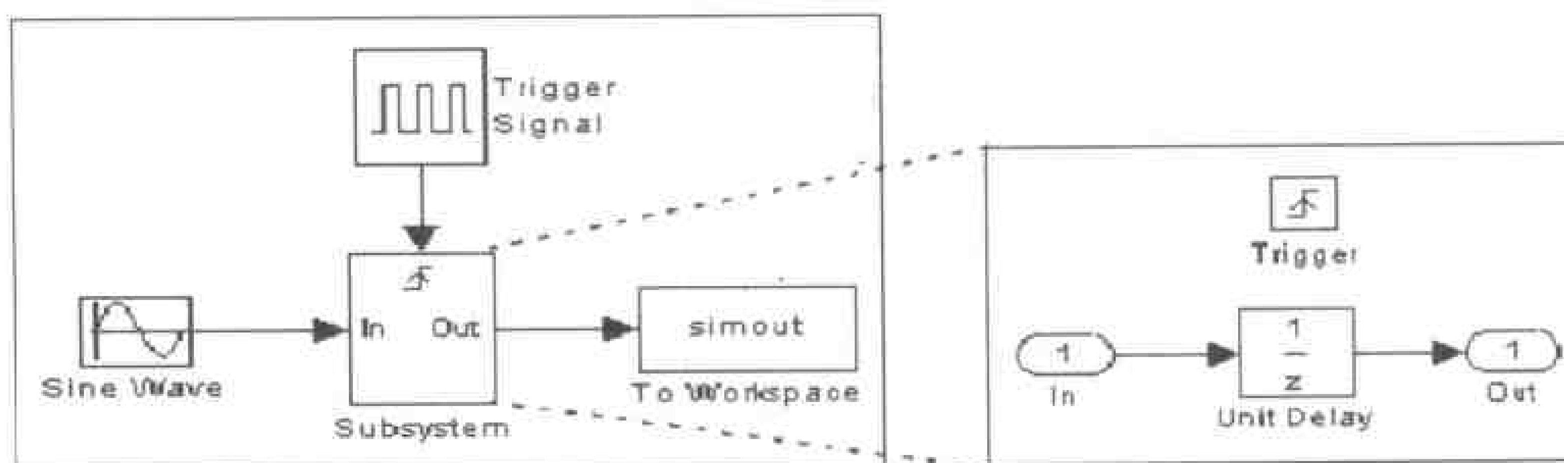


图 6.36　一个简单的触发子系统

系统，Simulink 自动在 Subsystem 模块的图标上加一个触发符号和一个控制信号输入端口，如图 6.37 所示。

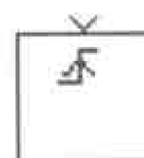


图 6.37　触发符号和控制端口

选择触发类型，打开 Trigger 模块对话框，如图 6.38 所示，并为 Trigger type 参数选择一个值：

- rising：在触发信号向正的方向过零时，触发发生。
- falling：在触发信号向负的方向过零时，触发发生。
- either：在触发信号向任意一个方向过零时，触发发生。

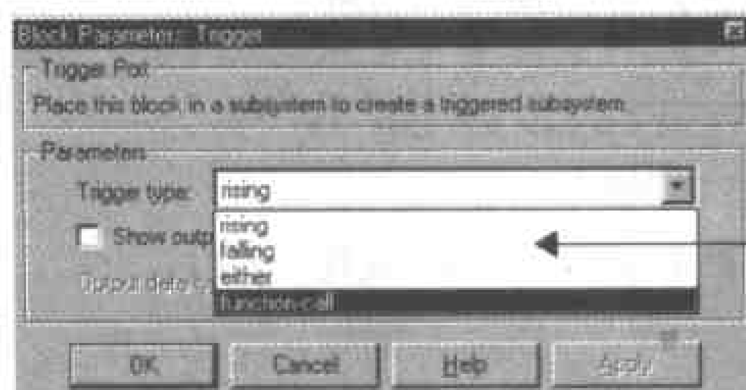


图 6.38　Trigger 模块对话框及说明

Simulink 在 Trigger 和 Subsystem 模块图标上使用不同的符号来指示上升沿触发、下降沿触发，及上升和下降沿同时触发（图 6.39）。



图 6.39　子系统模块上的触发符号

（1）触发事件之间的输出和状态。与使能子系统不同的是，在触发事件之间，触发子系统总是保持上一次触发的输出值。在触发再次发生时，也不重置子系统的状态。任意离散模块的状态总是在两个触发事件之间转换。

（2）触发控制信号输出。Trigger 模块对话框中也有一个 Show output port 复选框（见图

6.40)，用于对输出触发控制信号进行控制。

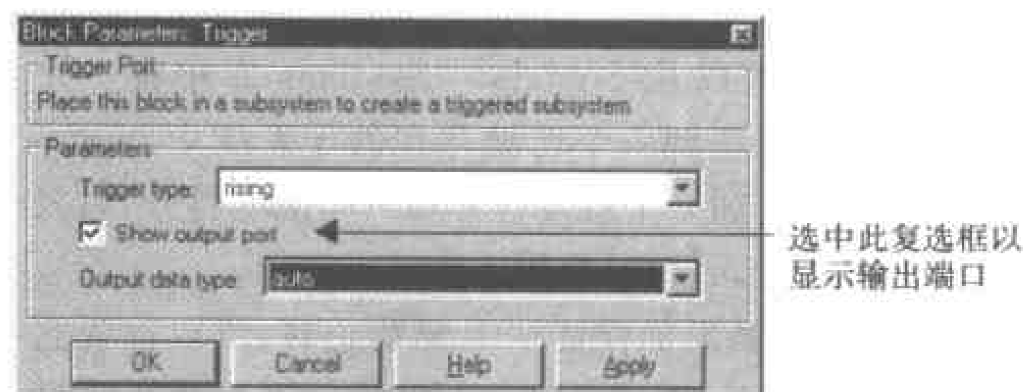


图 6.40　显示输出端口复选框

在 Output data type 栏指定输出数据类型，可以是 auto，int8 或 double。auto 项使数据类型与连接端口的数据类型(int8 或 double)自动保持一致。

2. 函数调用子系统

在 Trigger type 栏下还有一个选项：function-call。可以创建一种触发子系统，它的执行不是由信号值决定，而是由 S-function 的内部逻辑决定的。这样的子系统被称作函数调用子系统。有关函数调用子系统的更多信息，参看有关 S-function 的参考书。

6.4.3　触发子系统可以包含的模块

在仿真期间，触发系统只在特定的时间执行。因此，适合用于触发子系统的模块有：

- 继承采样时间的模块，如 Logical Operator(逻辑操作符)模块、Gain(增益)模块等。
- 采样时间设置为－1 的离散模块，这表示它的采样时间是从驱动模块继承的。

6.5　触发加使能子系统

第三种条件执行子系统是前两种条件执行子系统类型的合并。这种子系统叫做触发加使能子系统，它是使能子系统和触发子系统的一个组合子系统，如图 6.41 所示。

一个触发加使能子系统包含一个使能输入端口和一个触发输入端口。当触发事件发生

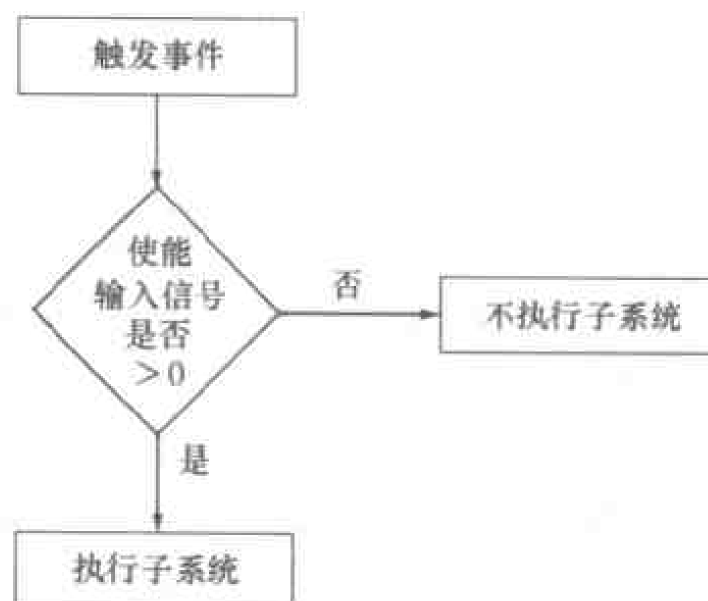


图 6.41　触发加使能子系统类型

时，Simulink 检查使能输入端口，如果它的值大于零，则执行该子系统。

1. 创建触发加使能子系统

创建触发加使能子系统，可以从 Signals& Systems 库中将一个 Trigger 模块和一个 Enable 模块拖到一个已经存在的子系统中，Simulink 在 Subsystem 模块的图标上加一个触发符号和一个使能符号，以及触发和使能的控制信号的两个输入端口（见图 6.42）。

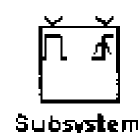


图 6.42 触发、使能符号及其控制端口

与使能子系统一样，当一个触发加使能子系统禁止时，可以设置它的输出值。详见 6.4.1“创建一个使能子系统”中的“在子系统禁止时设置输出值”。同样，可以指定子系统再次激活时的状态值。详见“设置子系统激活时的状态”。

分别给 Enable 和 Trigger 模块设置参数。设置过程与前面所述的单独的模块设置过程一样。

2. 一个触发加使能子系统示例

如图 6.43 所示的模型可以作为触发并使能子系统的一个简单的例子。

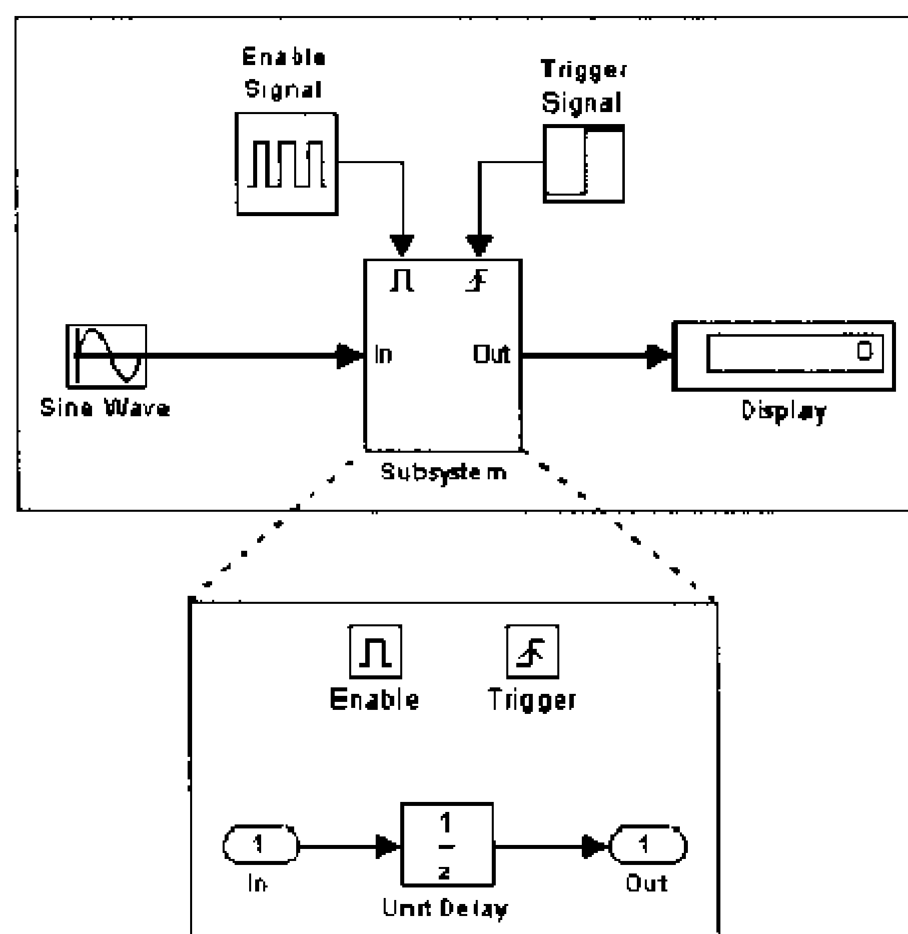


图 6.43 触发加使能子系统示例

3. 创建交替执行子系统

将条件执行子系统与 Merge 模块（参看第 7 章中的 Merge 模块）组合使用，就可以创建一系列实用模型的当前状态而交替执行的子系统。例如，图 6.44 为一种使用两个 Enable 模块和一个 Merge 模块组成的电流变换器模型，模拟将交流电转化为“直流”的设备。

在这个例子中，标签为 pos 的模块在交流电流的波形为正时被激活，即执行，它把波形无变化地传到输出口。标签为 neg 的模块是当波形为负时执行，它将该波形的负向部分反向。Merge 模块将当前使能（执行）的模块的输出传送给 Mux 模块，然后将该波形与原始交流波形一同输出给 Scope 模块，显示波形如图 6.45 所示。



图 6.44　包含两个 Enable 和一个 Merge 模块的电流变换器模型



图 6.45　模型输出波形

6.6　Simulink 仿真原理

本节面对已经掌握了 Simulink 基本方法的读者，介绍的内容涉及 Simulink 仿真的原理、方法及特点。当需要建立一个复杂的模型，而且有可能是多采样率或混合系统时，请仔细阅读本节。

6.6.1　Simulink 工作程序

一般而言，一个 Simulink 模型内的模块包含下列特性：输入矢量，u；输出矢量，y；状态矢量，x，如图 6.46 所示。

状态矢量可能由连续状态、离散状态或两者混合构成。这些量之间的数学关系可以用下



图 6.46 模块特性

列方程表示：

输出：$y = f_0(t, x, u)$

下一输入：$x_{d_{k+1}} = f_u(t, x, u)$

导数：$x'_c = f_d(t, x, u)$

其中：$x = \begin{bmatrix} x_c \\ x_{d_k} \end{bmatrix}$

仿真的过程由初始化和仿真两个阶段组成。在初始化阶段：

- 模块的参数传给 MATLAB 计算。其结果作为模块的实际参数值。
- 不具有条件执行的子系统被其中所含的模块替代。
- 模块分类排序以便随时更新。分类算法构建一个表，如具有直通输出的模块在驱动模块更新以后才会被更新。正是在此步骤，才探测代数环。参看后面“代数环”。
- 检验模块间的连接以确保某模块输出的矢量长度满足被驱动模块对输入的要求。

至此，仿真已准备运行。模型的仿真是采用数值积分法。配备 ODE 的仿真器(仿真算法)取决于模型提供其连续状态的求导能力。计算导数分两个步骤：首先，根据在分类阶段测定的顺序计算每个模块输出；然后，每个模块根据时间、输入和状态数计算导数。计算得到的导数矢量传递给仿真器，仿真器将用该矢量计算下一个时间点的新的状态矢量。新的状态矢量一经计算出，提供数据的模块和 Scope 模块就随之更新。

1. 过零检测

Simulink 使用过零检测去探测持续信号的间断点。过零检测在如下方面起十分重要的作用：

- 状态事件的处理；
- 不连续信号的精确积分。

(1) 状态事件的处理。当一个状态事件值的改变导致一个系统发生明显的变化时，就说该系统经历了一个状态事件。例如撞击地面的跳球是一个简单的状态事件。在使用一种变步仿真器仿真这样一个简单系统时，仿真器根本模仿不了跳球接触地面时间步。这样导致的计算结果是跳球几乎总是越过撞击点(越标)，造成穿透地面。

Simulink 利用过零检测以保证时间步(在机器精度内)与状态事件发生的时间同步。由于时间步发生在撞击时刻，所以仿真不会产生越标，且从负速度到正速度的过渡是极其灵敏的(即在间断点处非圆角过渡)。读者不妨在 MATLAB 命令窗口键入 bounce，可见一个跳球的演示程序。

(2) 非连续信号的积分。计算数值积分的程序是根据积分信号是连续和具有连续导数的假设而设计的。若在一个积分步中遇到一个不连续点(状态事件)，Simulink 将使用过零检测探测间断点发生的时刻，在间断点的左侧加一个时间步。然后，跨过间断点并开始下一个连续

信号段的积分步。

(3) 执行过程。使用过零检测的 Simulink 模块的例子之一是 Saturation 模块。Saturation 模块的过零检测探测状态事件包括：

- 输入信号到达上限；
- 输入信号离开上限；
- 输入信号到达下限；
- 输入信号离开下限。

定义自身状态事件的 Simulink 模块被认为具有内置过零检测。若用户需要一个过零检测事件的确切标志信息，应使用 Hit Crossing 模块。参看本节“内置过零检测模块”中的列表。

探测一个状态事件取决于一个内部过零检测信号。对这个信号模块框图不能使用。对 Saturation 模块而言，用于探测上限的过零检测信号为 zcSignal = UpperLimit − u，其中，u 是输入信号。

过零检测信号有一个方向属性，它可有以下的值：

- rising：过零检测事件发生在一个信号上升到或通过零点，或在信号离开零点并为正值时；
- falling：过零检测事件发生在一个信号下降到或通过零点，或在信号离开零点并为负值时；
- either：过零检测事件发生在上升或下降条件出现时。

对于模块的上限，过零检测的方向是 either。这能够使用相同的过零检测信号探测进入和脱离饱和事件。

若允许误差过大，Simulink 的过零检测有可能失败。例如，设一个过零检测事件发生在某一个时间步内，但时间步开始和结束值并未表明符号发生改变，则仿真器将越过“过零点”而没有将其检出。

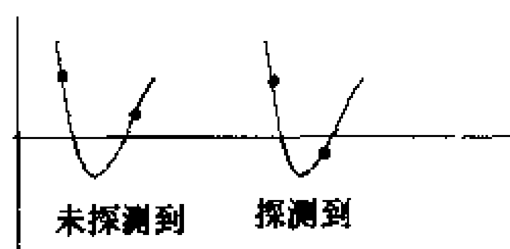


图 6.47 过零检测示意

图 6.47 显示了一个过零的信号。在左图中，积分器“越过”过零检测事件。在右图中，仿真器探测到了事件。

若用户担心这种情况发生，需减小允许误差以保证仿真器采取足够小的时间步。

编者提示：可能会构建一个间断点高频波动（震动）模型。这种仿真系统实际上是不可实现的，如小质量弹簧。因为震动造成重复出现同方向的过零，所以仿真的时间步就变得过小，这样会使仿真处于暂停状态。

若用户担心在模型中可能出现这种情况，可以通过选中 Simulation Parameters 对话框的 Diagnostics 选项上的 Disable zero crossing detection 复选框来禁止过零检测。虽然禁止过零检测可以减少问题的出现，但用户将不能从过零检测带来的精度改善上获得好处。最好的办法是找出模型中可能出现此类问题的根源。

(4) 带有过零检测功能的模块，见表 6.6。

表 6.6　带有过零检测功能的模块

模块名	过零检测描述
Abs	一个:探测输入从上升或下降方向过零的时刻
Backlash	二个:一个探测上阈值起作用的时刻;另一个探测下阈值起作用的时刻
Dead Zone	二个:一个探测进入死区的时刻(输入信号减去下限);另一个探测脱离死区的时刻(输入信号减去上限)
Hit Crossing	一个:探测输入跨过阈值的时刻。这种过零检测不受 Simulation Parameters 对话框的 Disable zero crossing detection 复选框的影响
Integrator	若置位端口存在,探测置位发生的时刻。若输出受限,则有三个过零检测:一个到达上饱和限的时刻;一个探测到达下饱和限的时刻;一个探测脱离饱和的时刻
MinMax	一个:对输出矢量的各元素,探测一个输入信号为最小或最大值的时刻
Relay	一个:若转换处于“off”状态,探测 switch on 点;若转换处于“on”,探测 switch off 点
Relational Operator	一个:探测输出变化的时刻
Saturation	二个:一个探测到达或离开上限的时刻;一个探测到达或离开下限的时刻
Sign	一个:探测输入通过零点的时刻
Step	一个:探测阶跃时刻
Subsystem	对条件执行子系统而言:一个用于可能存在的使能端口,一个用于可能存在的触发端口
Switch	一个:探测转换条件发生的时刻

2. 代数环

有些 Simulink 模块有直通输出输入端口。这是指:在不知进入模块输入端口的信号值情况下,这些模块的输出不能被计算。一些有直通输出输入端口的模块如下:

- Elementary Math 模块;
- Gain 模块;
- Integrator 模块的初始条件端口;
- Product 模块;
- State-Space 模块,在存在非零矩阵 D 时;
- Sum 模块;
- Transfer Fcn 模块,当分子和分母的阶数相等时;
- Zero-Pole 模块,当零点数与极点数相等时。

欲知一个模块是否为直通输出,请参看第 7 章各个模块的特性表。

(1) 代数环系统。当一个直通输出输入端口由相同模块的输出(直接或通过其他直通输出模块的反馈)驱动时,就会产生代数环,如图 6.48 所示(反例,请参看后面“非代数直通输出环”)。

准确地说，这个环意味着 Sum 模块的输出是处于强制等于第一个输入 u 减 z（即 $z = u - z$）的状态 z。其解 $z = u/2$，但大多数代数环的解不可能一眼就可看出。很容易建构一个含有多个代数状态变量 $z1, z2, \ldots$ 等的矢量代数环。如图 6.49 所示的模型：

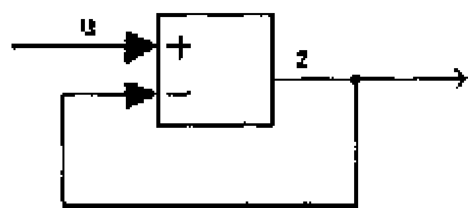


图 6.48 一个简单代数环示例

Algebraic Constraint 模块（见第 7 章"Algebraic Constraint 模块"）提供一种建构代数方程组模型和指定初始推测值的方便

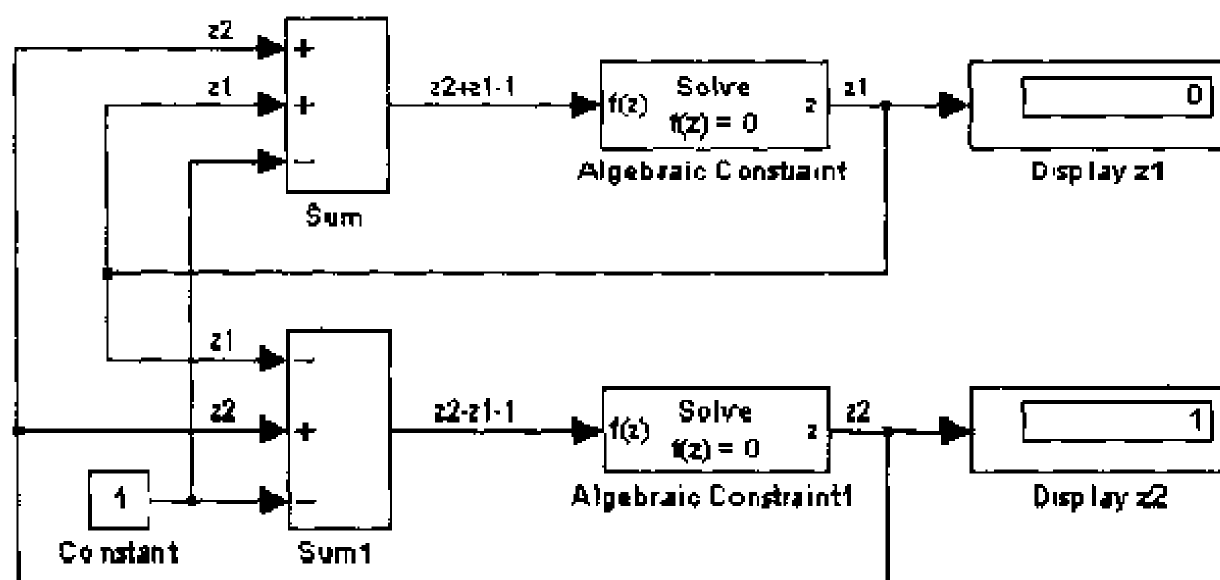


图 6.49 含多个代数环的模型

途径。Algebraic Constraint 模块强制其输入信号 $F(z)$ 为零，并输出一个代数状态 z。该模块输出必须使输入为零的值，这个输出通过某个反馈途径必须影响输入。用户可以在模块对话框中输入一个代数状态的初始推测值以提高代数环仿真器的效率。

一个标量代数环代表一个标量代数方程或约束项 $F(z) = 0$，其中 z 是环中某个模块的输出，函数 F 由从环中其他模块到这个模块输入的反馈路径组成。如前面的单模块的例子，$F(z) = z - (u - z)$。而对于上面的矢量环，有方程组：

$$\begin{cases} z2 + z1 - 1 = 0 \\ z2 - z1 - 1 = 0 \end{cases}$$

代数环起因于一个模型中包含了一个代数约束项 $F(z) = 0$。这种强制可能出现在用户正在建立的系统进行互连的过程中，或是当用户正在调试一个(DAE)系统模型时，也可能产生。

当一个模型包含了一个代数环，Simulink 将在每一个时间步调用一个环解释程序。如可能，环仿真器进行迭代运算对可能出现的问题求解。这样一来，含有代数环的模型，其运行速度要比没有的慢。

为求 $F(z) = 0$，Simulink 的环仿真器将牛顿法（weak line search 和 rank-one updates）用于一个偏导数雅柯比矩阵。虽然这种方法是可行的，但因为没有状态 z 的初始推测值，有可能使得环仿真器不能收敛，从而产生了代数环。用户可以通过将一个 IC 模块（通常用于指定信号的初始条件）置于代数环的一条线上，以便为该线指定一个初始推测值。正如图 6.49 所示，也可以使用一个 Algebraic Constraint 模块为代数环里的一条线指定一个初始推测值。

只要可能，应使用一个 IC 模块或一个 Algebraic Constraint 模块用于指定一个代数环状态变量的初始估计。

(2) 非代数直通输出环。实际上，并非所有包含直通输出模块的环路都是代数型的。如：

- 包括触发子系统的环路；
- 积分器输出到置位端口的环路。

对于一个触发子系统，仿真器能可靠地确保子系统的输入在触发时刻是稳定的。这就允许将前一个时间步的输出用于计算当前时间步的输入。因此，也就排除了一个代数环仿真器。

考虑图 6.50 所示的例子。

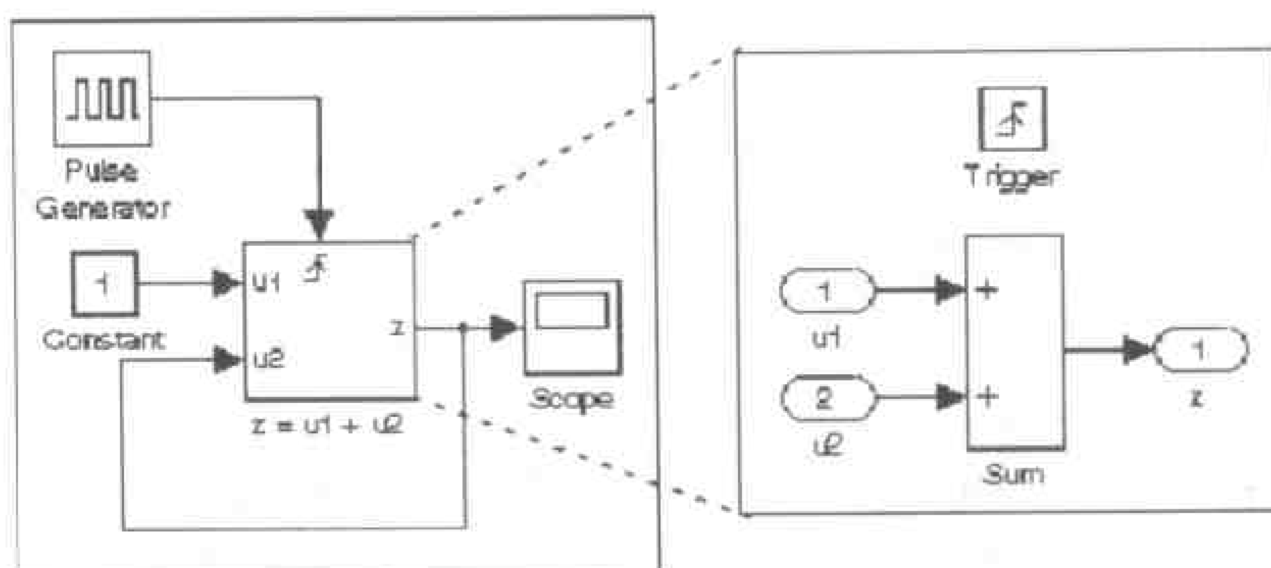


图 6.50 稳定系统举例

该系统的求解方程为：

$$z = 1 + u$$

其中，u 为最近一次子系统被触发时的 z 值。系统输出为一个阶梯函数，如图 6.51 所示。

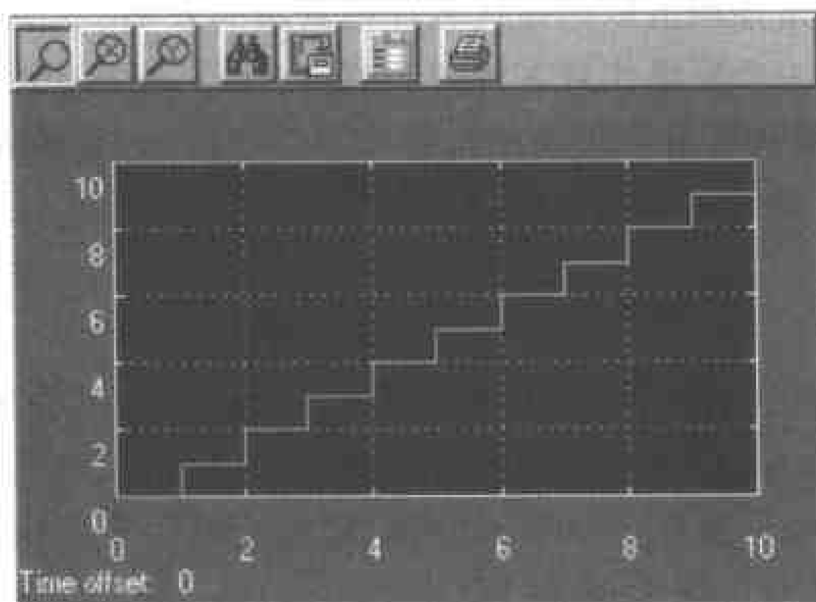


图 6.51 输出波形

下面考虑从上例中把触发端口取消后的影响。

在此情况下，加法器子系统 u2 端口的输入等于当前时间步子系统的输出。该系统的数学表达式为：

$$z = z + 1$$

可以看出：方程在数学上是无解的。

3. 不变常数

仿真时，模块必须有确定的采样时间。采样时间可以通过(系统本身或用户)定义，或是自动从驱动模块或被驱动模块继承。

例如，Constant 模块有无限长的采样时间，也就是具有一个恒定的采样时间。如果其他的模块接收的信号均来自同一个 Constant 模块，并且不从另外的模块继承采样时间的话，这些模块也是具有恒定采样时间的。这就是说，在仿真期间，除非模型的用户确实更改了参数，否则这些模块将不会改变输出。

例如，在图 6.52 所示的模型中，Constant 和 Gain 模块都具有恒定采样时间。

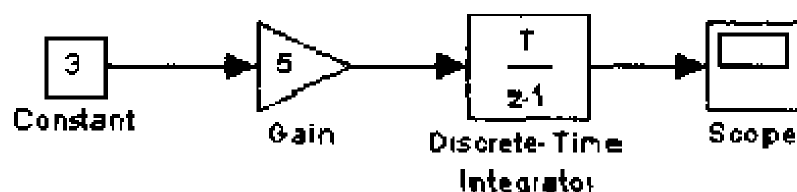


图 6.52　恒定采样时间的传递

因为 Simulink 支持在仿真期间对模块参数进行修改，所以，所有模块(包括定采样时间的模块)在模型的有效采样时间产生输出。由此，所有模块在每一个采样时间期间计算输出；而对于纯连续系统，则在每一个仿真时间步计算输出。因为在一次仿真期间，定采样时间模块的参数是不变的，所以，在仿真期间计算这些模块只会降低仿真的速度。

用户可以设置 Invariant Constants 参数以便将所有定采样时间模块从仿真“环”中除去。其作用体现在两点：一个是这些模块的参数在仿真期间不能改变；二是提高了仿真速度。至于速度能提高多少，取决于模型的复杂程度、含有定采样时间模块的数量以及仿真的实际采样率。

用户可以输入下面的指令设置用户模型的参数：

```
set_param('model_name', 'InvariantConstants', 'on')
```

也可以通过命令将参数赋值“off”以关闭属性。

用户可以通过 Format 菜单的 Sample Time Colors 控键将模块确定为定采样时间。定采样时间模块的颜色是红紫色。

6.6.2　离散时间系统注意事项

Simulink 不仅能仿真连续系统也能仿真离散系统。所以实际的模型不仅可以是多重采样速率的(也就是说模型可由不同采样时间的模块组成)；也可以是混合型的，即可以包含离散和连续的模块。

1. 离散模块

每一个离散模块都有一个针对输入的内置采样器和一个针对输出的零阶保持器。当离散模块与连续模块混合在一个模型中时，离散模块的输出在一次采样周期内保持恒定。只有当下一次采样后才更新。

2. 采样时间

Sample time 参数用于设置一个离散模块更新时的采样时间。通常，采样时间为一个标量变量。然而，也可通过在该栏内指定一个两元素的矢量以指定时间偏置量。

例如,将 Sample time 参数设置为矢量[Ts,offset],即将采样时间赋值给 Ts;偏置赋值给 offset。则离散模块只在采样时间和偏置为整数倍时才更新。

$$t = n * \mathrm{Ts} + \mathrm{offset}$$

其中:n 是一个整数,而偏置可以是正数或负数,但必须小于采样时间。

在仿真运行期间(包括暂停),用户不能更改一个模块的采样时间。若用户想要更改,必须停止仿真、更改并重新启动仿真,这样更改才起作用。

3. 纯离散系统

完全由离散模块组成的系统就是纯离散系统。从原理上讲可以使用任何一种仿真器对纯离散系统进行仿真,其解也不应该有任何差别。如果为了能得到产生的输出点数与采样数一致,则应选择离散仿真器。

4. 多重采样率系统

多重采样率系统由具有不同采样率的模块组成。当然也可以由离散模块或是离散和连续模块构建而成。例如,图 6.53 所示为一个简单的多重采样率离散模型。

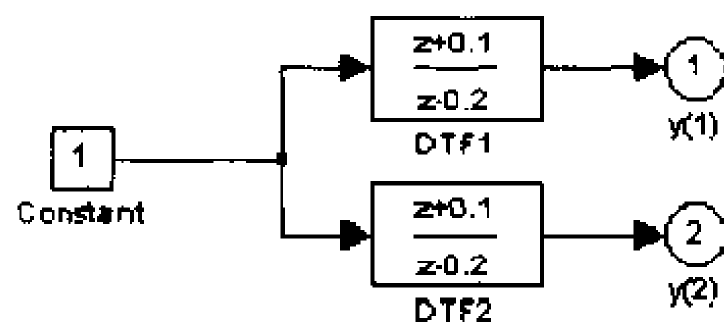


图 6.53 多制采样率模型

对于这个模型,DTF1 模块的 Sample time 被设置为[1 0.1](偏置为0.1)。DTF2 模块的 Sample time 被设置为 0.7(无偏置)。

启动仿真,然后在命令窗口使用函数命令:

```
>>[t,x,y] = sim('multirate', 3);
>>stairs(t,y)
```

绘出其输出图如图 6.54 所示。

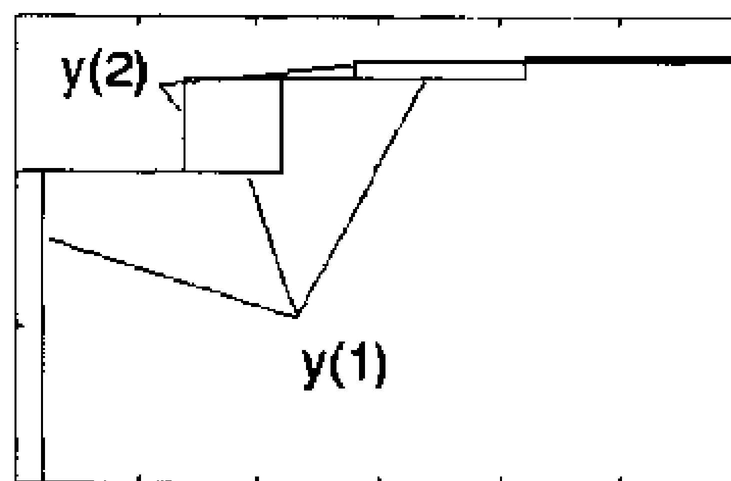


图 6.54 输出波形

对于 DTF1 模块,其偏置为 0.1,所以在 t=0.1 之前无输出。由于传递函数的初始条件为 0,则 DTF1 的输出 y(1)在此时间之前也为 0。

5. 采用时间的颜色特征

Simulink 通过采样时间的颜色特征用以鉴别一个模型中的不同采样率。不同的颜色表示不同的采样率,如表 6.7 所示。

表 6.7 采样时间的颜色特征

颜色	用途
黑	连续模块
洋红	Constant 模块
黄	混合(子系统分组模块,或 Mux 或 Demux 模块合并、分组信号,皆有不同的采样时间)
红	最快的离散采样时间
绿	次快的离散采样时间
蓝	第三快的离散采样时间
浅蓝	第四快的离散采样时间
深绿	第五快的采样时间
青色	触发采样时间
灰色	子步定点

熟悉 Simulink 的采样时间传播引擎(Sample Time propagation Engine,简写为 STPE)有助于理解这种特征。图 6.55 说明了一个采样时间 Ts 的 Discrete Filter 模块,它驱动一个 Gain 模块。由于 Gain 模块的输出仅仅是输入乘上一个常数,所以,其输出的采样率变化随 Discrete Filter 模块。换句话说,Gain 模块的实际采样率等于 Discrete Filter 模块的采样率。这就是 STPE 的基本原理。

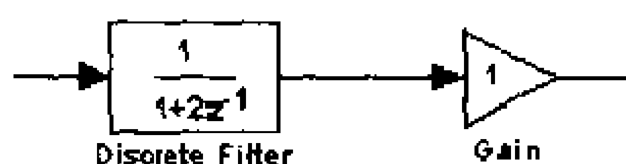


图 6.55 采样时间的传播

改变采样率:选中 Format 菜单的 Sample Time Colors,可以激活采样时间颜色特征。

Simulink 不会随用户更新模型而对其重新着色,所以,用户必须选择 Edit 菜单的 Update Diagram 确定更新模型的着色。再次选中 Sample Time Colors,禁止采样时间着色可以返回到原始色彩。

当使用采样时间色彩时,赋予一个模块的色彩取决于模块的采样时间和模型中其他模块的采样时间。

Simulink 根据如下规则设置各自模块的采样时间:

- 连续模块(如 Integrator,Derivative,Transfer Fcn 等模块)按定义为连续;
- 常数模块(例如 Constant 模块)按定义为常数;
- 离散模块(如 Zero-Order Hold,Unit Delay,Discrete Transfer Fcn 等模块)的采样时间须由用户在模块对话框中明确指定;
- 其他的模块根据其输入间接地定义采样时间。例如,在 Integrator 模块后的 Gain 模块就被认为是一个连续模块;而跟在 Zero-Order Hold 模块后面的 Gain 模块就被看做是其采样

时间与 Zero-Order Hold 模块采样时间相同的一个离散模块。

对于输入具有不同的采样时间的模块来说，如果所有采样时间是最快采样时间的整倍数，则模块就被赋予最快的输入采样时间。若当前仿真器为一个变步仿真器，模块就被赋予连续采样时间。若为定步仿真器，并且可以计算出采样时间的最大公约数（基频采样时间），则使用该采样时间；否则仍然使用连续采样时间。

编者提示：Mux 和 Demux 模块仅仅是组分控制器，所以通过模块的信号保留原有的定时信息。因此，若 Demux 模块的输入源具有不同的采样时间，则从模块发出的连线可以是不同颜色的。在这种情况下，Mux 和 Demux 模块的颜色编码为"混合"（黄），以表示模块处理的是多制采样率信号。

类似地，包含不同采样时间模块的 Subsystem 模块也是黄色的，因为与它们关联的信号采样率不是单一的。如果一个于系统内的所有模块以一个采样率运行，则 Subsystem 模块就着以对应该采样率的色彩。

在某些情况下，Simulink 也会将采样时间反向传递给源模块，但这必须在不影响仿真输出的情况下。例如，在图 6.56 所示的模型中，Simulink 认为 Signal Generator 模块正在驱动一个 Discrete-Time Integrator 模块，所以 Signal Generator 模块和 Gain 模块的采样时间被设置为 Discrete-Time Integrator 模块的采样时间。

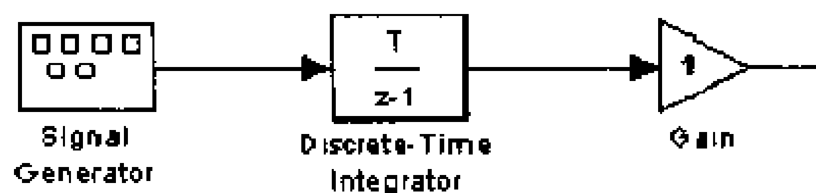


图 6.56　采样时间的反向传播

用户可以通过激活 Sample Time Colors 并注释所有模块为红色来指定这种情况。因为 Discrete-Time Integrator 模块仅仅关注其输入的采样时间，所以，这种更改不会影响仿真结果，但肯定会改善性能。

将 Discrete-Time Integrator 模块用一个连续 Integrator 模块替换（如图 6.57 所示），并选中 Edit 菜单的 Update Diagram 对模型进行重新着色，这样，Signal Generator 模块和 Gain 模块就变为连续模块，其颜色也变为黑色。

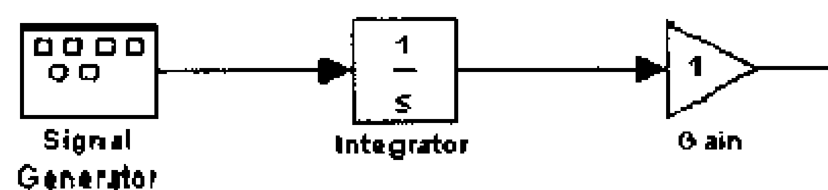


图 6.57　纯连续模型

6. 连续和离散混合系统

连续和离散混合系统由连续和离散模块组成。这种系统可以使用任何一种算法进行仿真（某些方法的效率和精度可能高些）。但对大多数连续和离散混合系统而言，Runge-Kutta 变步法，ode23 和 ode45，在效率和精度上比别的方法更优越。由于离散模块的采样与保持带来的间断点，所以在连续和离散混合系统中最好不要使用 ode15s 和 ode113 法。

第7章 Simulink 库模块

Simulink 提供了十分丰富的模块库，每一类模块库包含许多库模块，每一种模块都有各自的特性和参数，可以完成相应的任务。本章将对所有的库模块进行介绍。

主要内容：Simulink 库模块及相关信息。

学习目的：通过本章的学习，了解库模块的用途、参数及其特性，为建立自己的仿真模型打好基础。

编者提示：虽然掌握本章的内容足以建立满足一般要求的动态系统仿真模型，但对于复杂的仿真模型或大型应用程序而言，还须了解模型参数、库模块的公用参数和特有参数，相关内容见附录。

7.1 Simulink 模块库

打开 Simulink 库浏览器窗口，就可找到 Simulink 模块库。模块按照不同用途归入相应的 Simulink 模块库中：

(1) 输入源(Sources)模块库，包含产生信号的模块。

(2) 接收器(Sinks)模块库，包含显示、写输出及用于匹配的模块。

(3) 离散系统(Discrete)模块库，包含描述离散时间量的模块。

(4) 连续系统(Continuous)模块库，包含描述线性函数的模块。

(5) 非线性系统(Nonlinear)模块库，包含描述非线性函数的模块。

(6) 数学运算(Math)模块库，包含描述一般数学函数的模块。

(7) 常用函数和查表(Functions & Tables)模块库，包含描述一般函数和查表运算的模块。

(8) 信号和系统(Signal & Systems)模块库，包含可进行多路传输或多路分解、实现内/外间输入/输出、向模型其他部分传递数据、创建子系统、实现其他用途的模块。

(9) 模块集和工具箱(Blocksets and Toolboxes)库，包含系统定制的一些额外模块库。

(10) 演示程序(Demos)库，包含一些非常有用的 MATLAB 和 Simulink 的演示程序。

由于篇幅的限制，本章只对前 8 类进行介绍。

7.2 库模块预览

下面按照库的分类，给出 Simulink 3.0 版本的所有库模块。为查阅方便，各个表只给出

了模块名及其用途的简单说明，详细描述及用法会在后面各节里介绍。

表 7.1 输入源(Sources)模块

模块名	用 途
Band-limited White Noise	在线性系统中加入带限白噪声
Chirp Signal	产生 Chirp 信号
Clock	显示和提供仿真时间
Constant	产生一常数，可由参数对话框指定
Digital Clock	产生以指定采样周期的仿真时间
Discrete Pulse Generator	产生矩形脉冲
From File	读取一个文件中的数据
From Workspace	读取存在于工作空间的一个矩阵中的数据
Pulse Generator	产生矩形脉冲
Ramp	产生一个上升或下降斜坡信号
Random Number	产生正态分布的随机信号
Repeating Sequence	将指定的时限信号延拓为周期信号
Signal Generator	可产生正弦波、方波和锯齿波
Sine Wave	产生正弦波
Step	产生阶跃函数信号
Uniform Random Number	产生均匀分布的随机信号

表 7.2 接收(Sinks)模块

模块名	用 途
Display	显示输入值
Scope	显示仿真过程中产生的信号
Stop Simulation	终止非零输入时的仿真
To File	将输入数据存入文件
To Workspace	将输入数据写入工作空间，并存入指定变量的矩阵
XY Graph	用 MATLAB 图形显示两个信号的关系

表 7.3 离散系统(Discrete)模块

模块名	用 途
Discrete Filter	实现离散滤波器(IIR 和 FIR)
Discrete State - Space	实现离散状态空间系统
Discrete - Time Integrator	实现离散时间积分

续表 7.3

模块名	用　　途
Discrete Transfer Fcn	实现离散传递函数
Discrete Zero - Pole	实现离散零-极点传递函数
First - Order Hold	实现一阶采样保持
Unit Delay	将采样保持延迟一个周期
Zero - Order Hold	实现一个采样周期的零阶保持

表 7.4　连续系统(Continuous)模块

模块名	用　　途
Derivative	时间微分
Integrator	积分
Memory	将前一步的输入整体延迟一个步长后作为当前步的输出
State - Space	线性状态方程
Transfer Fcn	线性传递函数
Transport Delay	按指定时间对输入做延迟
Variable Transport Delay	按第二输入指定的时间对第一输入(信号)做延迟
Zero - Pole	零-极点传递函数

表 7.5　数学运算(Math)模块

模块名	用　　途
Abs	求输入的绝对值或模
Algebraic Constraint	强制输入信号为零
Combinatorial Logic	建立逻辑真值表
Complex to Magnitude - Angle	求输入的幅值、相角
Complex to Real - Imag	求输入的实部和虚部
Dot Product	求两个输入的点积
Gain	用指定的常数、变量或表达式同输入相乘
Logical Operator	对输入进行指定的逻辑运算
Magnitude - Angle to Complex	将幅值和相角输入转换为复数信号输出
Math Function	对输入进行常用数学运算
Matrix Gain	用矩阵同输入相乘
MinMax	求输入的最小值或最大值
Product	求输入的乘积或商
Real　Imag to Complex	将实部和虚部输入转换为复数信号输出
Relational Operator	对两个输入进行比较
Rounding Function	取整

续表 7.5

模块名	用　途
Sign	指出输入的符号,输出 1,0 或 −1
Slider Gain	在仿真过程中允许用滑动条改变增益
Sum	求和
Trigonometric Function	三角函数运算

表 7.6　函数和表(Functions & Tables)模块

模块名	用　途
Fcn	提供 C 语言标准的表达式对输入进行运算
Look-Up Table	对输入进行分段线性映射
Look-Up Table (2−D)	对两个输入进行分段线性映射
MATLAB Fcn	提供 MATLAB 函数或表达式对输入进行运算
S-Function	访问一个 S-Function

表 7.7　非线性系统(Nonlinear)模块

模块名	用　途
Backlash	模拟回差控制函数
Coulomb & Viscous Friction	模拟零点不连续的线性增益函数
Dead Zone	规定零输出的区域
Manual Switch	两输入之一选通
Multiport	两模块输入之一选通
Quantizer	以指定阶梯函数量化输入
Rate Limiter	限定输入的变化量
Relay	两常数输出之一选通
Saturation	限定信号的范围
Switch	按指定阈值对输入之一选通

表 7.8　信号和系统(Signal & Systems)模块

模块名	用　途
Bus Selector	输出选通信号
Configurable Subsystem	调用用户指定模块
Data Store Memory	定义共享数据存储区
Data Store Read	从共享数据存储区读数据
Data Store Write	将数据写入共享数据存储区
Data Type Conversion	转换信号的数据类型

续表 7.8

模块名	用途
Demux	把矢量输入分解成标量或分矢量输出
Enable	给子系统加一使能端
From	把 Goto 模块的输入信号作为输出
Function-Call Generator	函数调用子系统
Goto	将输入信号传递给不同的 From 模块
Goto Tag Visibility	定义 Goto 模块标签的显示范围
Ground	使模块的输入端接地(未接输入时必须如此)
Hit Crossing	探测输入是否达到所设阈值
IC	设置输出信号的初始条件
Inport 或 In1	为子系统或外输入创建一个导入端口
Merge	将多个输入合并为一个标量输出
Model Info	显示模型中的版本控制信息
Mux	将标量或分矢量合成为一个矢量输出
Outport 或 Out1	为子系统或外输出创建一个导出端口
Probe	输出输入信号的宽度、采样时间以及信号类型
Subsystem	创建子系统
Terminator	用于末端模块输出端的匹配
Trigger	给子系统加一触发控制端口
Width	输出输入矢量信号的宽度

7.3 库模块相关说明

在介绍各种模块之前,有几点说明,希望读者留意。

本章中所有涉及的模块,按照模块的分类库(从 7.4 节至 7.11 节)及模块的英文字母顺序排列介绍。介绍内容包括两个部分:

1. 各个模块库所含模块的信息表

表中给出了每一个模块的信息:

(1) 模块名、图标;

(2) 模块用途描述;

(3) 模块参数;

(4) 模块特性。关于模块特性有以下几点提请注意:

① 数据类型(Data Type)。给出了模块对输入(包括参数)及输出信号的要求。

② 直通输出(Direct Feedthrough)。若列出,则表明模块或其端口是直通输出(即当模块的输入值未知时,模块不计算输出)的。欲知有关直通输出更多的信息,参看 6.5 节中的"代数

环"。

③ 采样时间(Sample Time)。模块的采样时间由模块本身或由其驱动模块或被驱动模块决定。欲获得更多的信息,参看6.5节中的"采样时间"。

④ 标量扩展(Scalar Expansion)。标量被扩展为矢量的过程称为标量扩展。某些模块可以将输入和参数进行适当的扩展。欲获得更多的信息,参看"3.3.14 输入和参数的标量扩展"。

⑤ 状态数(States)。离散和连续状态的数量。

⑥ 矢量化(Vectorized)。模块接受或产生矢量信号,详情参看3.3.13"矢量输入和输出"。

⑦ 过零检测(Zero Crossings)。探测状态事件。欲获得更多的信息,参看6.5节中的"过零检测"。

2. 编者提示

我们根据模块的实际应用,在每一个模块库中选择了一些较为典型的问题进行详细的介绍,以供读者使用参考。

7.4 输入源库模块

输入源模块库包含多种常用的信号和数据发生器。由于可以通过模块提供的对话框设置控制参数,所以可以满足大多数要求。表7.1列出了所有输入源模块的名称和用途以供读者查阅使用。

7.4.1 Band-Limited White Noise 模块

1. 功能描述

产生正态分布随机数,用于连续或混合系统。本模块与Random Number模块的主要区别在于本模块以指定的采样时间产生输出并与噪声的相关时间有关。从理论上讲,连续白噪声具有0相关时间,常数值的功率谱密度(PSD)以及无穷大的协方差。如果当噪声与物理系统的固有带宽有极小的相关时间,在理论近似的容许范围内,我们就可以认为噪声是白噪声,这是很有用的,但实际上,任何现实系统是不可能以白噪声分布的。

在Simulink上,用户可以使用具有比系统最小时间常数更小的相关时间的随机序列来模拟白噪声的效果。Band-Limited White Noise模块就可以产生这样的序列。噪声的相关时间就是模块的采样率。采用比最快的动态系统小得多的相关时间来进行较为精确的仿真,为了得到好的结果,可以通过指定相关时间:

$$t_c = \frac{1}{100}\frac{2\pi}{f_{max}}$$

f_{max}(rad/s)——系统的带宽。为产生合适强度的噪声,噪声方差的运算应反映出从一个连续的功率谱密度到一个离散的噪声方差的固有变换。采用噪声的相关时间 t_c 作为换算因子,保

证了一个连续系统对我们需要近似模拟的白噪声的响应具有与系统本身相同的方差。对于这种换算，Band-Limited White Noise 模块输出信号的方差不同于在 Noise power (intensity)[噪声功率谱(密度)]对话框中的参数。该参数实际上是白噪声 PSD 的幅度。当白噪声的方差为无穷大，模块产生的近似就是模块输出信号的方差，为(Noise Power)/t_c。

2. 参数与对话框

Band-Limited White Noise 模块的参数对话框如图 7.1 所示。

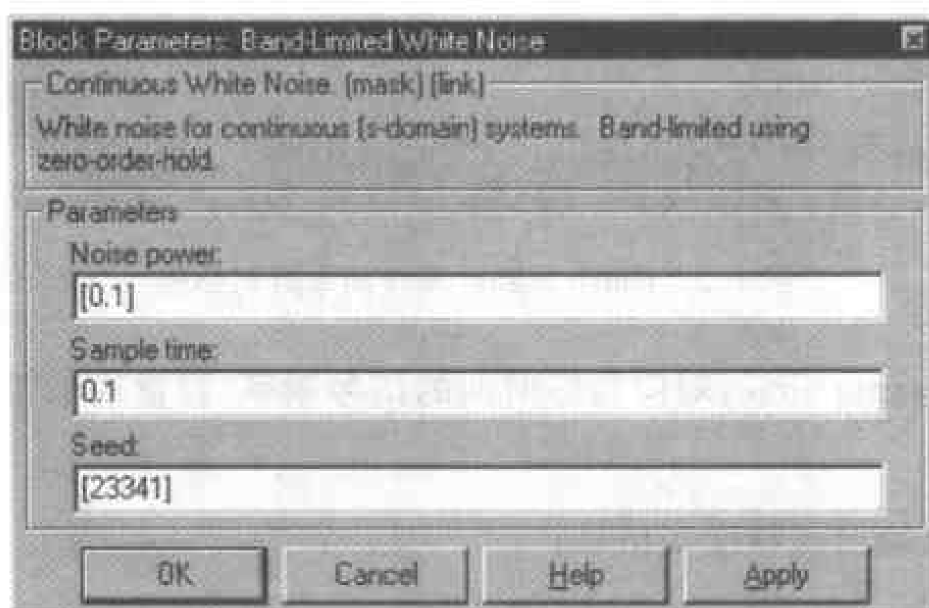


图 7.1 Band-Limited White Noise 模块的参数对话框

- Noise power：白噪声 PSD 的幅度，默认值为 0.1；
- Sample time：噪声的相关时间，默认值为 0.1；
- Seed：随机数信号发生器的初始种子，默认值为 23341。

3. 特性

- 数据类型：模块输出双精度实数；
- 采样时间：离散；
- 标量扩展：Noise power 和 Seed 参数及输出；
- 矢量化。

7.4.2 Chirp Signal 模块 

1. 功能描述

Chirp Signal 模块产生频率随时间线性增加的正弦信号(调频信号)。此模块可用于非线性系统的谱分析。该模块产生标量或矢量输出。

2. 参数和对话框

Chirp Signal 模块的参数对话框如图 7.2 所示。

- Initial frequency：信号的初始频率，指定为标量或矢量值。默认值为 0.1Hz。
- Target time：频率达到 Frequency at target time 参数值时的时间，标量或矢量值。在此时间之后，频率以相同的变化速率变化。默认值为 100 s。

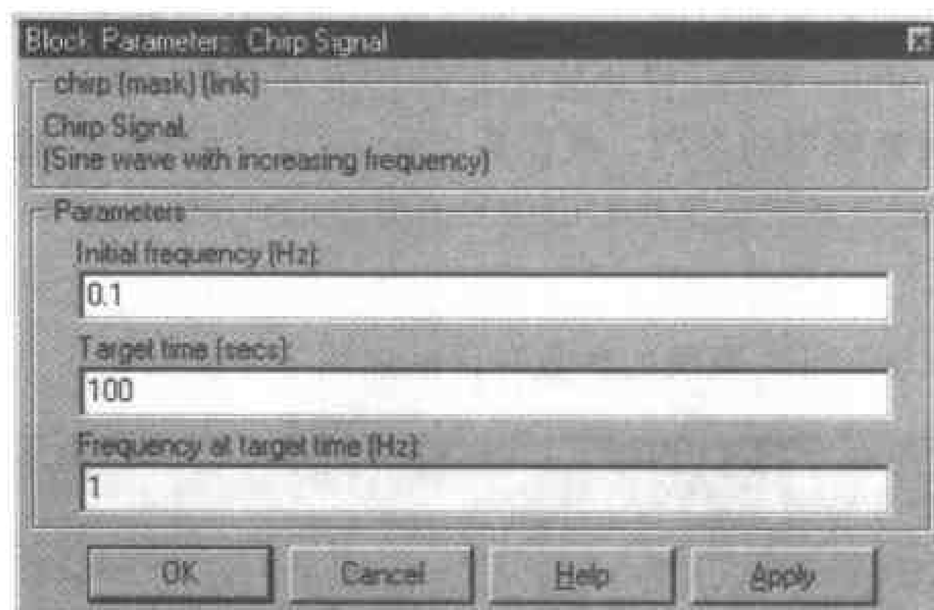


图 7.2 Chirp Signal 模块的参数对话框

- Frequency at target time：对应目标时间的信号频率，标量或矢量值。默认值为 1Hz。

3. 特性

- 数据类型：模块输出双精度实信号；
- 采样时间：连续；
- 标量扩展：所有参数；
- 矢量化。

7.4.3 Clock 模块

1. 功能描述

Clock 模块在每一仿真步输出当前仿真时间。该模块对其他需要仿真时间的模块非常有用。如果需要离散系统的通用时间，请用 Digital Clock 模块。

2. 参数和对话框

Clock 模块的参数对话框如图 7.3 所示。

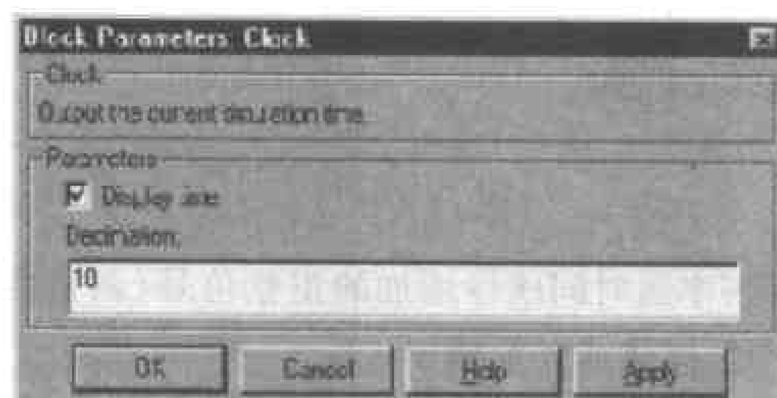


图 7.3 Clock 模块的参数对话框

- Display time：选中 Display time 复选框，显示在 Clock 模块图标内的当前仿真时间，然后有下面的参数显示。

• Decimation：Decimation 参数值随时钟不断更新而增加，可以是任意正整数。例如，当抽取率为 1 000 时，对于 1ms 的固定积分步长，时钟将在 1s、2s……进行更新。

3. 特性

• 数据类型：Clock 模块输出双精度实信号；
• 采样时间：连续。

7.4.4 Constant 模块 [1]

1. 功能描述

Constant 模块输出与时间无关的实或复数。它只有一个输出，用户可通过对话框上的 Constant value 参数栏指定常数（或数列），其长度也就决定了是标量或矢量。

2. 参数和对话框

Constant 模块的参数对话框如图 7.4 所示。

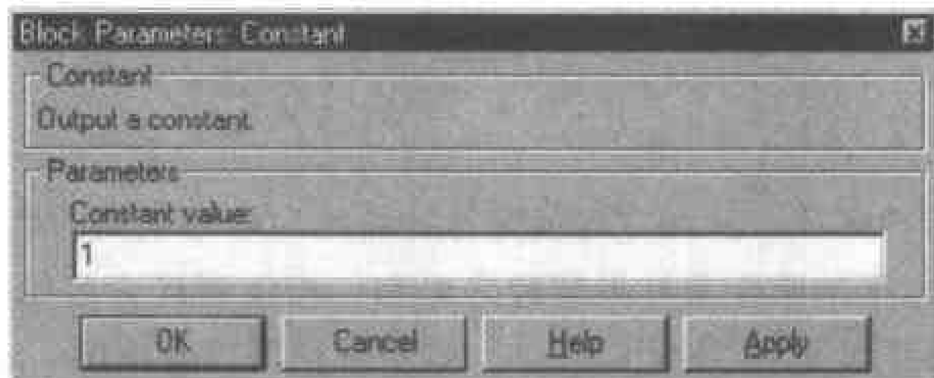


图 7.4 Constant 模块的参数对话框

• Constant value：模块的输出。默认值为 1。

3. 特性

• 数据类型：模块输出是数值型复或实信号，并由 Constant value 的值决定；
• 采样时间：常数；
• 可矢量化。

7.4.5 Digital Clock 模块 [12:34]

1. 功能描述

Digital Clock 模块仅仅在指定的采样间隔内输出仿真时间。在其他时间，输出保持前一次的值不变。若用户需要离散系统的通用时间，请选用本模块而不是 Clock 模块（输出连续时间）。

2. 参数和对话框

Digital Clock 模块的参数对话框如图 7.5 所示。

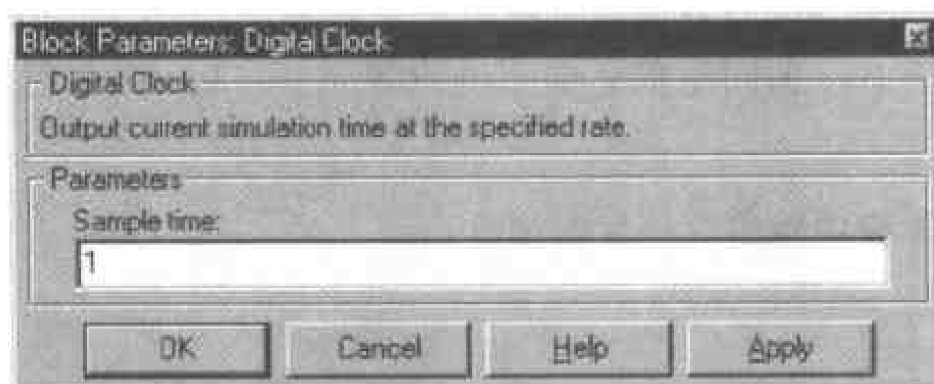
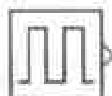


图 7.5　Digital Clock 模块的参数对话框

- Sample time：采样时间。默认值为 1s。

3. 特性

- 数据类型：Digital Clock 模块输出双精度型实信号；
- 采样时间：离散。

7.4.6　Discrete Pulse Generator 模块 

1. 功能描述

模块产生等间隔脉冲序列。脉冲宽度就是脉冲持续高电平期间的数字采样周期数。脉冲周期乃脉冲持续高、低电平的数字采样周期数之总和。相位延迟为起始脉冲所对应的数字采样周期数。相位延迟可正可负，但不能超过一个数字采样周期数相位。采样时间必须大于 0。

本模块可用于离散或混合系统。如需要连续性脉冲序列，请选用 Pulse Generator 模块。

2. 参数和对话框

Discrete Pulse Generator 模块的参数对话框如图 7.6 所示。

- Amplitude：脉冲幅度。默认值为 1。

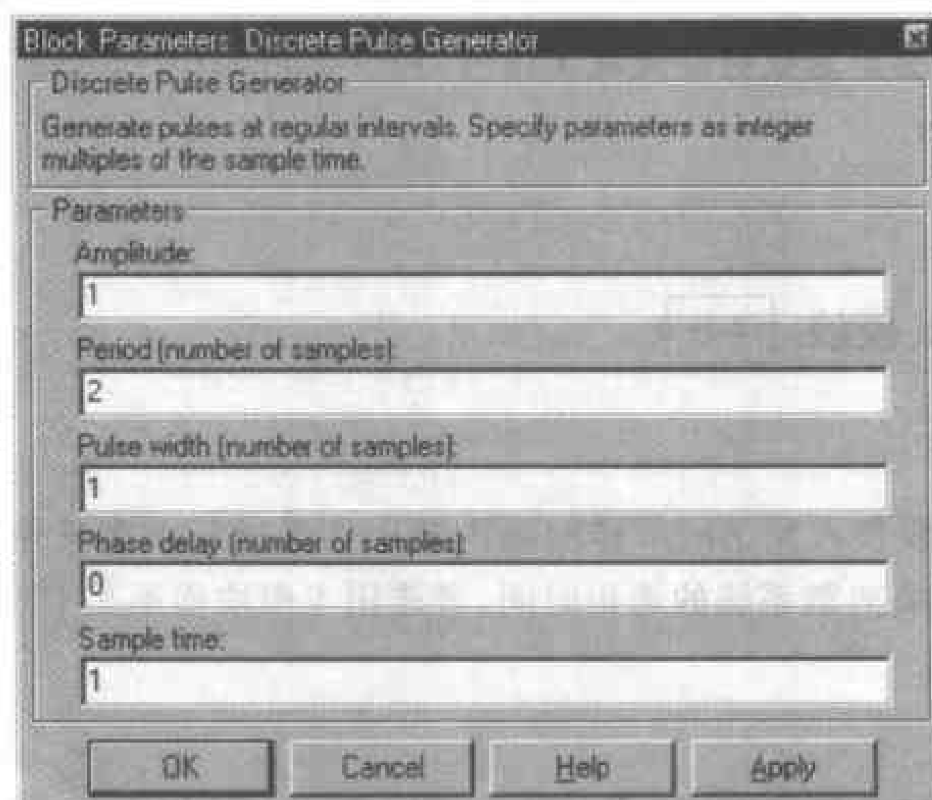


图 7.6　Discrete Pulse Generator 模块的参数对话框

- Period:脉冲数字采样周期。默认值为 2。
- Pulse width:脉冲持续高位的采样周期数。
- Phase delay:用采样数表示的一个脉冲产生前的延迟。默认值为 0。
- Sample time:采样周期。默认值为 1。

3. 特性

- 数据类型:输入和输出双精度实信号;
- 采样时间:离散;
- 标量扩展:参数;
- 可矢量化。

7.4.7 From File 模块 untitled.mat

1. 功能描述

From File 模块从一个指定的文件中读取数据并输出。模块图标显示提供数据的文件名。文件必须是不少于两行的矩阵。第一行必须是单调递增的时间,其他行的数据与第一行相应列上的时间一一对应。矩阵应具有如下形式:

$$\begin{bmatrix} t_1 & t_2 & \cdots & t_{final} \\ u1_1 & u1_2 & \cdots & u1_{final} \\ \vdots & \vdots & \vdots & \vdots \\ un_1 & un_2 & \cdots & un_{final} \end{bmatrix}$$

输出数据的宽度取决于被读文件的行数。模块用时间数据计算其输出,但不输出时间值。这意味着对于一个包含 m 行的矩阵,模块的输出为 $m-1$ 行的矢量,并由原矩阵中除第一行外的所有数据构成。

如果在某一时间需要一个输出,而这个时间又处在文件中两个时间点之间,则输出是将这两个时间点所对应的值经过线性内插法计算后得到的值。如果所需时间小于文件中第一个时间点或大于文件中最后一个时间点,Simulink 则使用最前两个点或最后两个点进行线性外插。

如果矩阵在相同时间点有大于或等于两列,则输出为所遇到的第一列的值。例如,某一个矩阵:

time values: 0 1 2 2

data points: 2 3 4 5

在时间 2,输出为 4,该值为所遇到的时间为 2 时的第一列数据。

Simulink 在仿真开始时将文件读入内存。这样用户就不能从在同一个模型中由 To File 模块命名相同的文件中读取数据。

编者提示:使用由 To File 或 To Workspace 模块保存的数据的方法及要求是,From File 模块可以不经任何修饰地读取一个由 To File 模块写入的数据。为了读取由 To Workspace 模块写入的数据并保存在一个文件中,要求:

① 数据必须包含仿真时间。仿真输出中含有时间数据的最简单的方法是在 Simulation Parameters 对话框的 Workspace I/O 栏中定义一个时间变量。欲知详情,请参看 5.3.2 节。

② 写入工作空间的数据形式不同于 From File 模块所期望的形式。在将数据存于文件之前，必须将其转换为 From File 模块要求的形式。

2. 参数和对话框

From File 模块的参数对话框如图 7.7 所示。

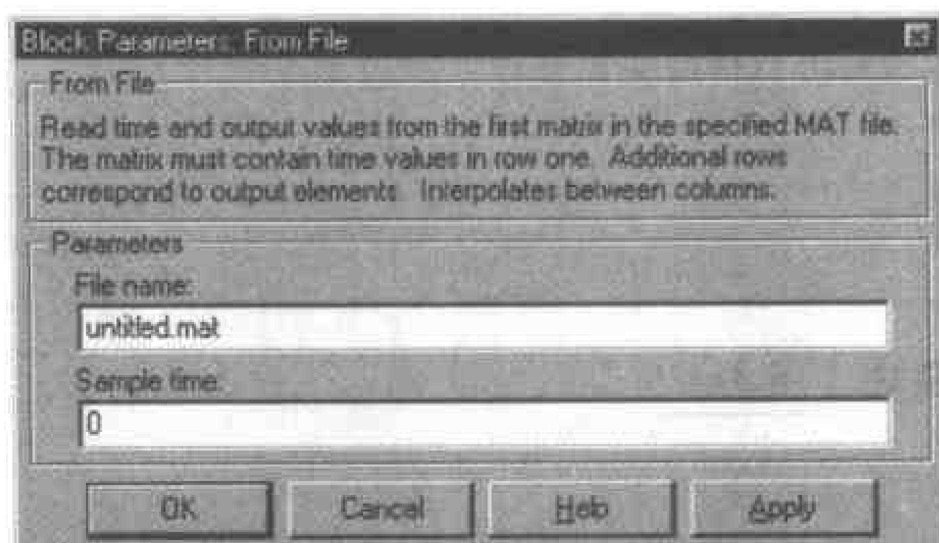


图 7.7　From File 模块的参数对话框

- **File name**：包含模块输入的数据的文件名。默认值为 untitled.mat。
- **Sample time**：从文件读数据的采样率。

3. 特性

- 数据类型：From File 模块输出双精度型实信号；
- 采样时间：由被驱动模块决定；
- 可矢量化。

7.4.8　From Workspace 模块 [T,U]

1. 功能描述

From Workspace 模块用于从 MATLAB 工作空间中读取数据。模块的 Data 参数通过求解一个矩阵或是包含信号和时间步的列表结构的 MATLAB 表达式来定义工作空间数据。矩阵或结构的格式与用于从工作空间载入数据的格式相同。参看第 4 章"从基本工作空间载入数据"。From Workspace 模块图标显示在 Data 参数栏内的表达式。

如果输入列表结构没指定输入值的对应时间，则每个值的对应时间就被假设为

$$t = (n-1) \times st$$

其中 n 为第 n 个输入值；st 表示模块的采样时间。

对应每一个时间步，From Workspace 模块输出取决于模块 Interpolate data 和 Hold final data value 参数的设置。表 7.9 归纳了对应不同的参数设置组合时的输出。

如果输入列表含有多个对应同一时间步的数据，则 Simulink 使用最后一个。例如，设输入列表为：

time：0 1 2 2
signal：2 3 4 5

在时间 2，输出为 5，即对应时间 2 的最后一个数据。

表 7.9 From Workspace 模块的输出组合

插值选项 (Intrp. Option)	保持选项 (Hold Option)	模块输出 (Block Output) ti < t < tf	模块输出 (Block Output) t > tf
选中	未选中	两相邻数内插	从最终数据外插
选中	选中	两相邻数内插	尾数据
未选中	未选中	最新数据	0
未选中	选中	最新数据	尾数据

编者提示：如果输出是结构(Steucture)或时间结构(Structure With Time)格式(参看"从基本工作空间载入数据")，则 From Workspace 模块可以直接读取一个 To Workspace 模块的输出(参看 To Workspace 模块)。如需要读取由某一个 To Workspace 模块写入到工作空间的矩阵，则要求矩阵必须添加一时间序列。

2. 参数和对话框

From Workspace 模块的参数对话框如图 7.8 所示。

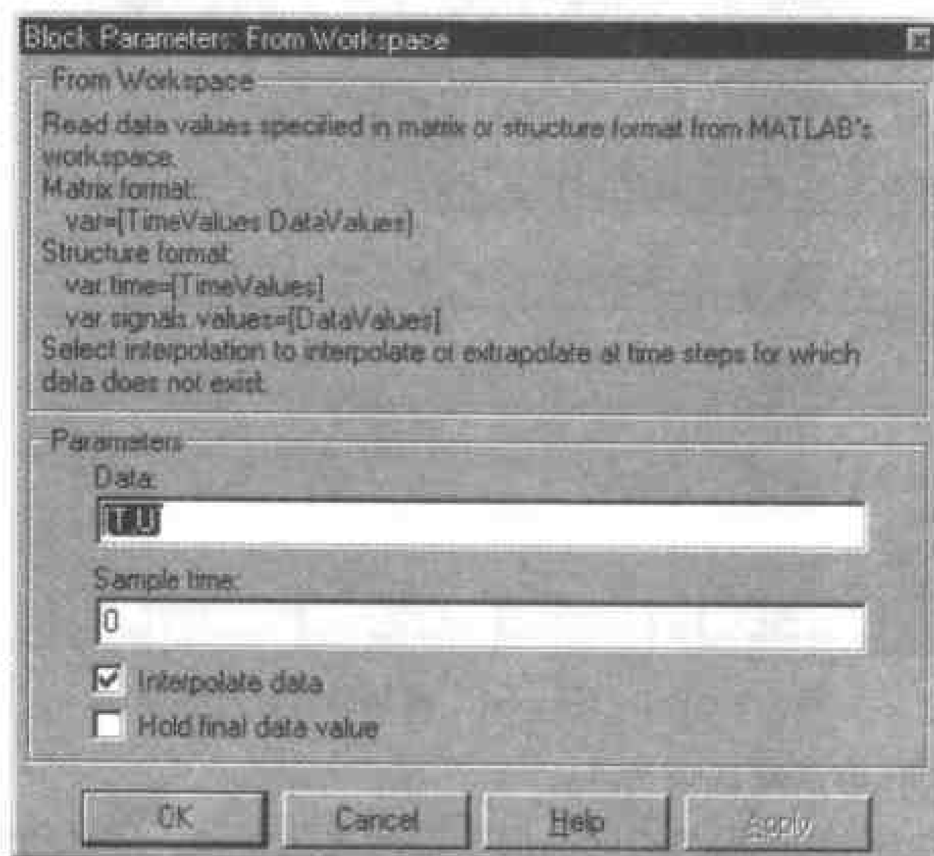


图 7.8 From Workspace 模块的参数对话框

• Data：用于计算一个矩阵或一个含有仿真时间和相应信号值的结构的表达式。例如，设工作空间含有一名为 T 的时间列矢量和一个名为 U 的对应信号矩阵，则该参数的默认表达式为[T，U]，产生一个含有要求输入列表的矩阵。如果要求的信号/时间矩阵或结构已在工作空间中，在该栏中输入结构或矩阵名即可。

• Sample time：从工作空间读取数据的采样时间。

- Interpolate data：该选项使模块在工作空间中无相应数据时的时间内进行线性内插计算（在 Hold final data value 参数为 off 时，则进行外插计算）。否则，当前输出为在有数据时的最新时间所对应的数据。
- Hold final data value：该选项使模块将最后一个有效数据作为输出。

3. 特性

- 数据类型：From Workspace 模块可输出任意数据类型的实或复信号；
- 采样时间：由被驱动模块决定；
- 可矢量化。

7.4.9 Pulse Generator 模块

1. 功能描述

Pulse Generator 模块产生固定频率脉冲序列。本模块适用于连续系统。若需产生离散脉冲，请使用 Discrete Pulse Generator 模块（参看“Discrete Pulse Generator 模块”）。

2. 参数和对话框

Pulse Generator 模块的参数对话框如图 7.9 所示。

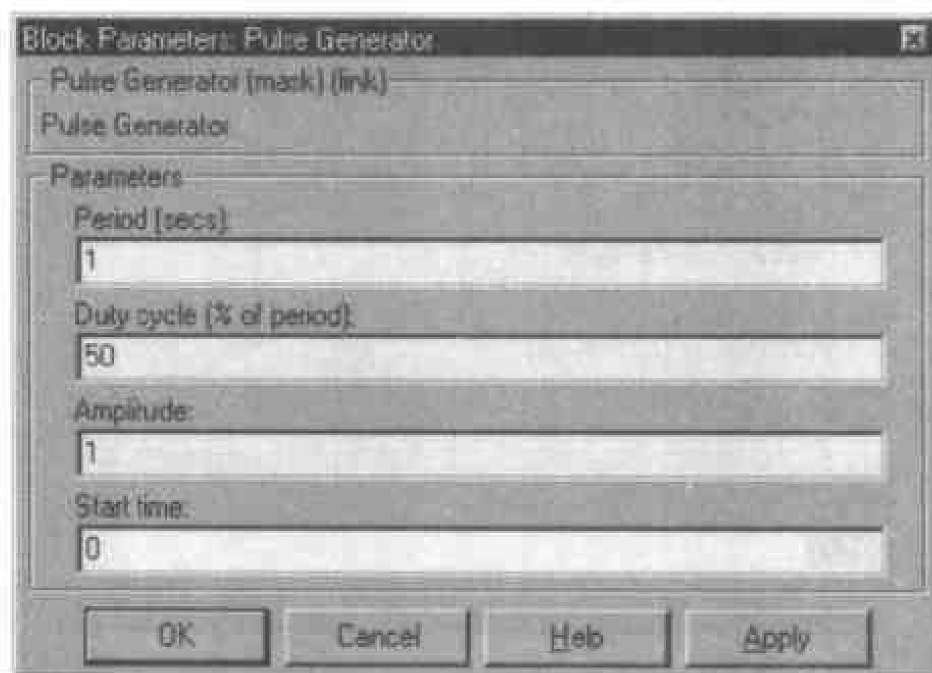


图 7.9　Pulse Generator 模块的参数对话框

- Period：脉冲周期(s)。默认值为 1s。
- Duty cycle：占空比：脉冲保持时间与间歇时间之比(%)。默认值为 50%。
- Amplitude：脉冲幅度。默认值为 1。
- Start time：产生脉冲前之延迟(s)。默认值为 0 s。

3. 特性

- 数据类型：本模块输出双精度型信号；
- 采样时间：继承；
- 标量扩展：参数；
- 可矢量化。

7.4.10　Ramp 模块

1. 功能描述

Ramp 模块可产生按指定初始时间、初始幅度和变化率的斜坡信号。

2. 参数和对话框

Ramp 模块的参数对话框如图 7.10 所示。

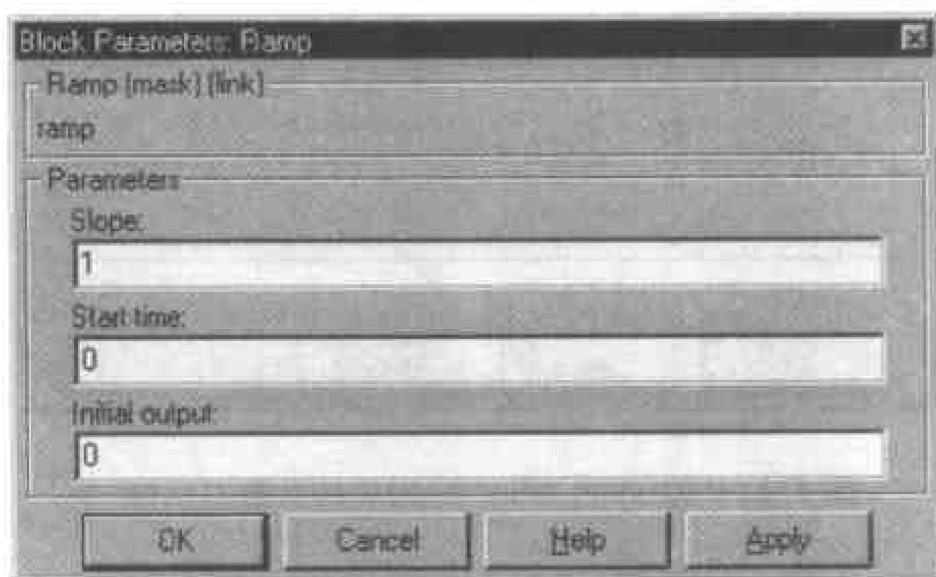


图 7.10　Ramp 模块的参数对话框

- Slope:输出信号的变化率。默认值为 1。
- Start time:信号的初始时间。默认值为 0。
- Initial output:信号的初始幅值。默认值为 0。

3. 特性

- 数据类型:输出双精度型信号;
- 采样时间:取决于驱动模块;
- 标量扩展;
- 可矢量化;
- 过零检测。

7.4.11　Random Number 模块

1. 功能描述

Random Number 模块产生正态分布的随机信号。在每一次仿真开始时,种子都设置为指定的值。默认时,产生的随机序列均值为 0 和方差为 1,即使用户改变这些参数。随机序列是可重复的并且可由相同种子和参数的任一 Random Number 模块产生。将 Initial seed 指定为矢量,模块可产生有相同均值和方差的一个随机信号矢量。

编者提示:若想获得均匀分布的随机信号,请使用 Uniform Random Number 模块。由于仿真器只能对相对平滑的信号进行积分,所以不能对由本模块产生及经过其传递的随机信号进行积分。若需要对随机信号进行积分,请使用 Band-Limited White Noise 模块作为信号源。

2. 参数和对话框

Random Number 模块的参数对话框如图 7.11 所示。

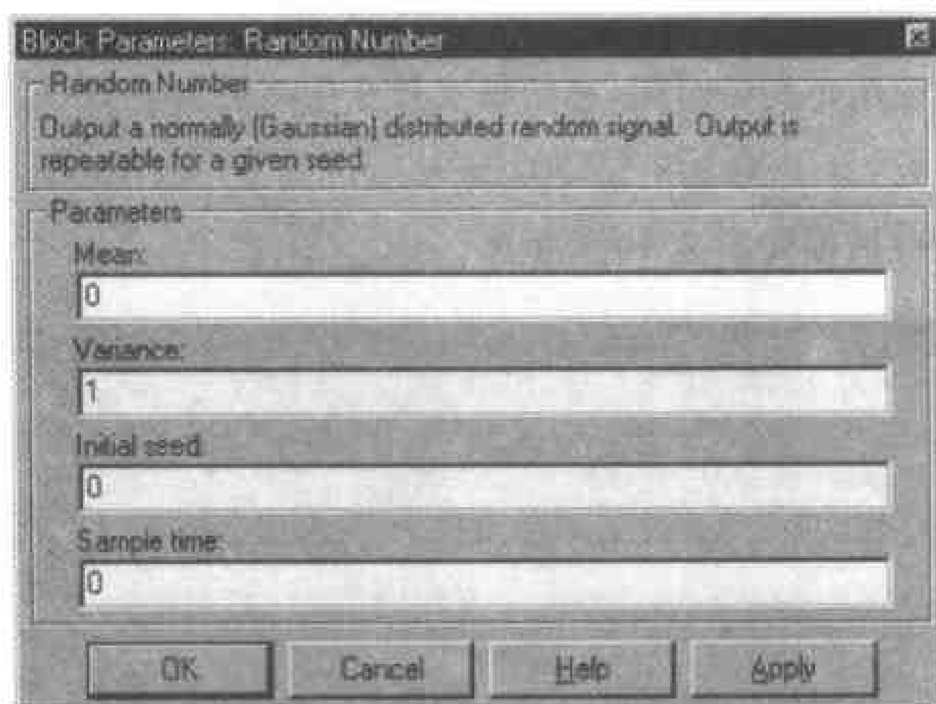


图 7.11 Random Number 模块的参数对话框

- Mean：随机信号的均值。默认值为 0。
- Variance：随机信号的方差。默认值为 1。
- Initial seed：随机信号发生器的初始种子。默认值为 0。
- Sample time：两次采样的时间间隔。默认值为 0，此时模块为连续采样时间。

3. 特性

- 数据类型：模块接受和输出双精度型信号；
- 采样时间：连续或离散；
- 标量扩展：参数；
- 可矢量化；
- 过零检测。

7.4.12 Repeating Sequence 模块

1. 功能描述

用户可通过本模块产生一个任意波形的周期信号。当仿真时间到达 Time values 矢量的最大时间值时，信号将重复。

本模块最好通过一维 Look-Up Table 模块的使用实现两点之间的线性插值。

2. 参数和对话框

Repeating table 模块的参数对话框如图 7.12 所示。

- Time values：一个单调递增的时间值矢量。默认值为[0 2]。
- Output values：输出值矢量。每一个值等于相同列的时间值。默认值为[0 2]。

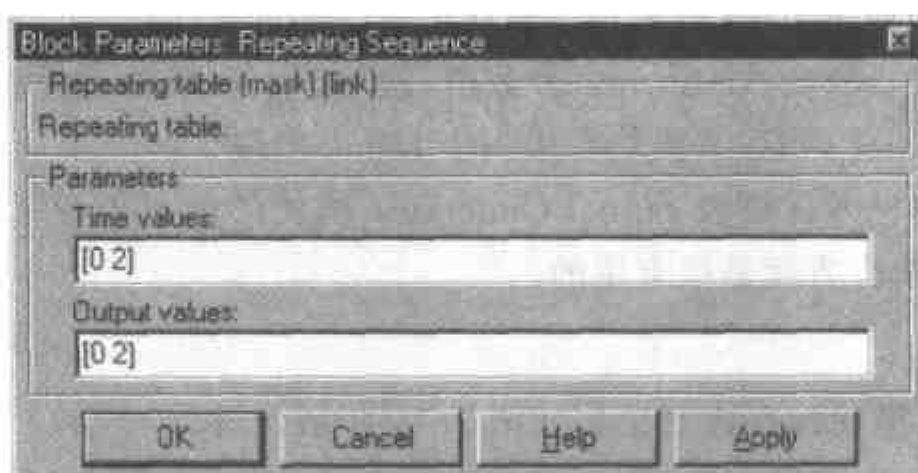


图 7.12　Repeating table 模块的参数对话框

3. 特性

- 数据类型：输出双精度型信号；
- 采样时间：连续。

7.4.13　Signal Generator 模块 

1. 功能描述

本模块可以产生三种不同波形的信号：正弦波、方波和锯齿波。信号的单位可以是 Hz（赫

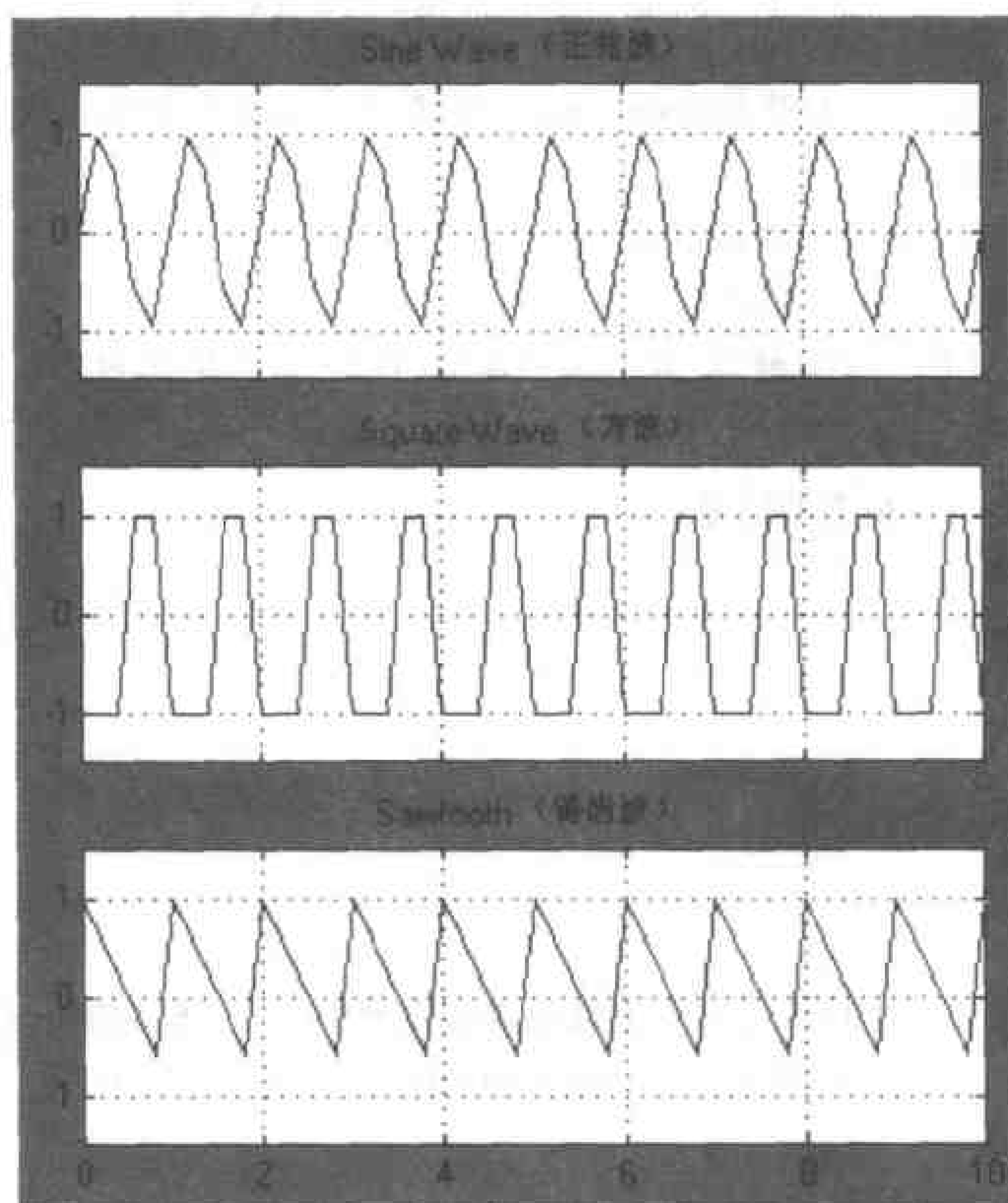


图 7.13　Signal Generator 模块产生的波形示例

兹)或 rad/s(弧度/秒)。图 7.13 给出了在默认值时 Scope 显示的信号波形。

编者提示:若指定一个负 Amplitude 参数值,就可产生一个 180°的相移。还可以通过不同的途径产生非 180°的相移,如在本模块后接一个 Delay 模块。

在进行仿真中,用户可以更改 Signal Generator 模块的输出设置。这在快速测定一个系统对不同类型输入的响应方面是很有用的。

2. 参数和对话框

Signal Generator 模块的参数对话框如图 7.14 所示。

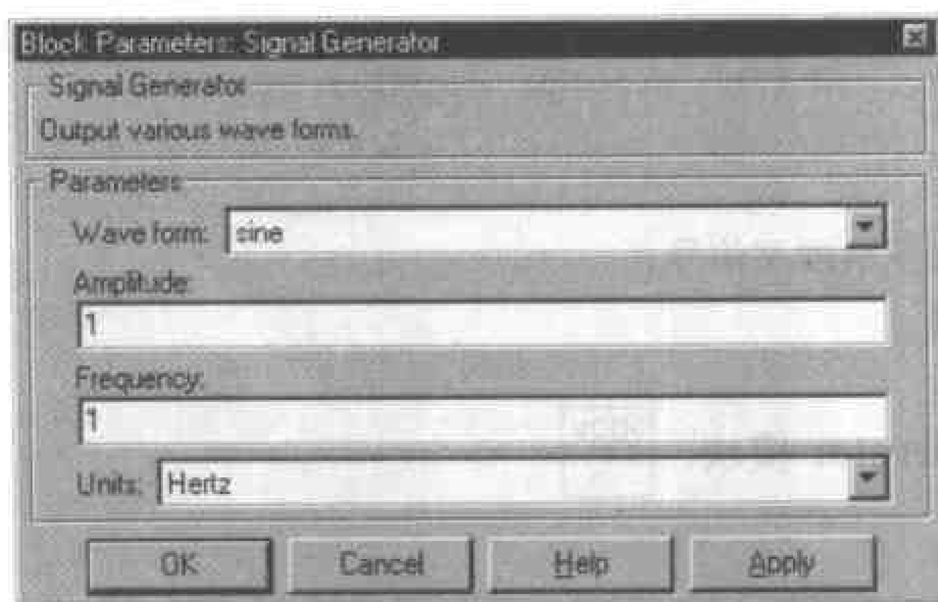


图 7.14　Signal Generator 模块的参数对话框

- Wave form:波形。正弦波(sine wave)、方波(square wave)或锯齿波(sawtooth wave)。默认值为正弦波。
- Amplitude:信号幅度。默认值为 1。
- Frequency:信号频率。默认值为 1。
- Units:信号的单位,Hz(赫兹)或 rad/s(弧度/秒)。默认单位为 Hz。

3. 特性

- 数据类型:输出双精度型信号;
- 采样时间:继承;
- 标量扩展:参数;
- 可矢量化。

7.4.14　Sine Wave 模块 

1. 功能描述

产生一个正弦曲线。本模块可以在连续或离散模型中工作。模块输出由下式决定:

$$y = \text{Amplitude} \times \sin(\text{frequency} \times \text{time} + \text{phase})$$

2. 参数和对话框

Sine Wave 模块的参数对话框如图 7.15 所示。

- Amplitude:信号幅度。默认值为 1。

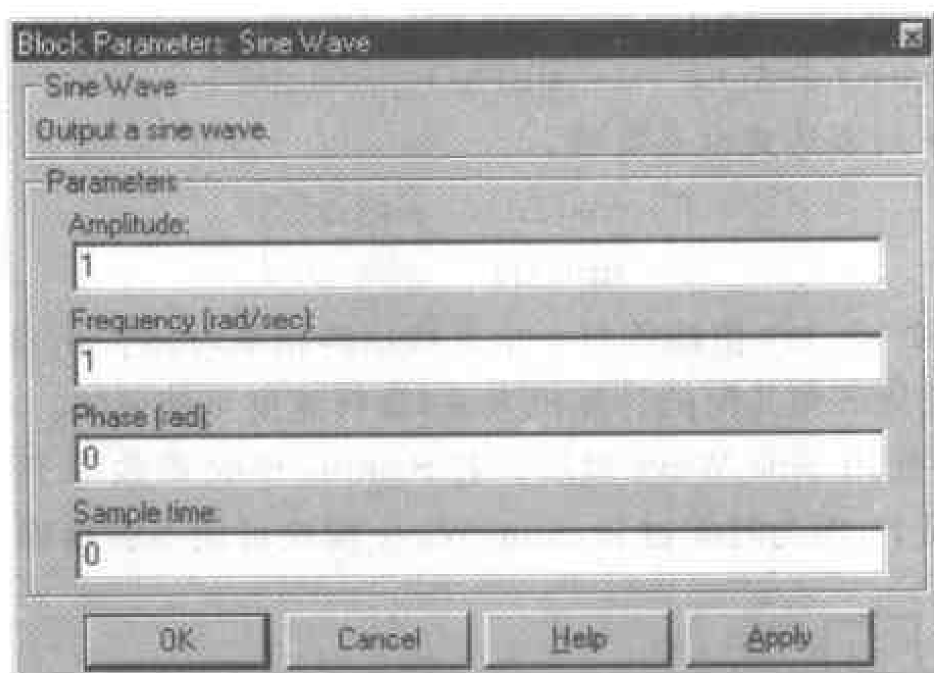


图 7.15　Sine Wave 模块的参数对话框

- Frequency：频率，单位 rad/s（弧度/秒）。默认值为 1rad/s。
- Phase：相移，单位 rad（弧度）。默认值为 0rad。
- Sample time：采样周期。默认值为 0。

3. 特性

- 数据类型：接受和输出双精度型实信号；
- 采样时间：连续，离散，或继承；
- 标量扩展：参数；
- 可矢量化。

4. 应用说明

（1）用户可以通过 Sample time 参数值的设置来决定模块为连续模型或离散模型工作：

- 0（默认值）模块工作在连续模型中；
- >0，模块工作在离散模型中；
- 1，模块工作在接受信号的模块模型中。

（2）在离散模型中应用 Sine Wave 模块时，Sample time 参数值大于 0，就好像模块正在驱动一个采样时间被设置为这个值的 Zero-Order Hold 模块。

在这种情况下使用 Sine Wave 模块允许用户构建带有纯数字正弦信号源的模型，而不是混合系统。混合系统本身是非常复杂的，因而仿真时间也长。

Sine Wave 模块采用一种增量算法而不是一种基于绝对时间概念的算法。因此，对于希望在不确定长度时间内运行的模型而言，本模块是有用的。如在震动或疲劳测试模型中。

所谓增量算法是根据在前一个采样时间计算的值计算正弦波。这种方法采用如下恒等式：

$$\sin(t+\Delta t) = \sin(t)\cos(\Delta t) + \sin(\Delta t)\cos(t)$$

$$\cos(t+\Delta t) = \cos(t)\cos(\Delta t) - \sin(\Delta t)\sin(t)$$

也可以把恒等式写成矩阵形式。

$$\begin{bmatrix} \sin(t+\Delta t) \\ \cos(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \cos(\Delta t) & \sin(\Delta t) \\ -\sin(\Delta t) & \cos(\Delta t) \end{bmatrix} \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$$

因为 Δt 是常数，所以下列表达式也是常数。

$$\begin{bmatrix} \cos(\Delta t) & \sin(\Delta t) \\ -\sin(\Delta t) & \cos(\Delta t) \end{bmatrix}$$

因此，这就成为一个 sin (t) 值矩阵与一个常数矩阵相乘，得到 sin $(t+\Delta t)$。这种算法对于那些没有硬件浮点支持三角几何的计算机来说，运行速度可能更快。

(3) 在连续模型中使用 Sine Wave 模块。若 Sample time 参数等于 0，则模块就工作在连续模型中。在这种情况下，因为时间过长，Sine Wave 模块可能不够精确。

7.4.15 Step 模块

1. 功能描述

Step 模块在指定时间产生一个可定义上、下电平的阶跃信号。若仿真时间小于 Step time 参数值，模块输出即为 Initial value 参数值。对于大于或等于 Step time，输出为 Final value 参数值。Step 模块根据参数的长度产生一个标量或矢量输出。

2. 参数和对话框

Step 模块的参数对话框如图 7.16 所示。



图 7.16　Step 模块的参数对话框

- Step time：从 Initial value 到 Final value 的阶跃时间，单位为 s。默认值为 1s。
- Initial value：仿真时间到达 Step time 时的模块输出。默认值为 0。
- Final value：仿真时间超过 Step time 时的模块输出。默认值为 1。
- Sample time：阶跃采样率。

3. 特性

- 数据类型：输出双精度型实信号；
- 采样时间：从被驱动模块继承；

- 标量扩展：参数；
- 可矢量化；
- 过零检测，探测阶跃次数。

7.4.16　Uniform Random Number 模块

1. 功能描述

Uniform Random Number 模块产生在整个指定时间周期内均匀分布的随机信号，信号的起始种子可由用户指定。起始种子在仿真开始时重新设置。产生的序列是可重复的，并且可以由任意具有相同起始种子和参数的 Uniform Random Number 模块产生。将 Initial seed 参数指定为矢量，可以产生矢量随机数序列。

编者提示：若需产生正态分布的随机序列，请使用 Random Number 模块。应避免对一个随机信号的积分，因为仿真器总是对相对平滑的信号进行积分。

2. 参数和对话框

Uniform Random Number 模块的参数对话框如图 7.17 所示。

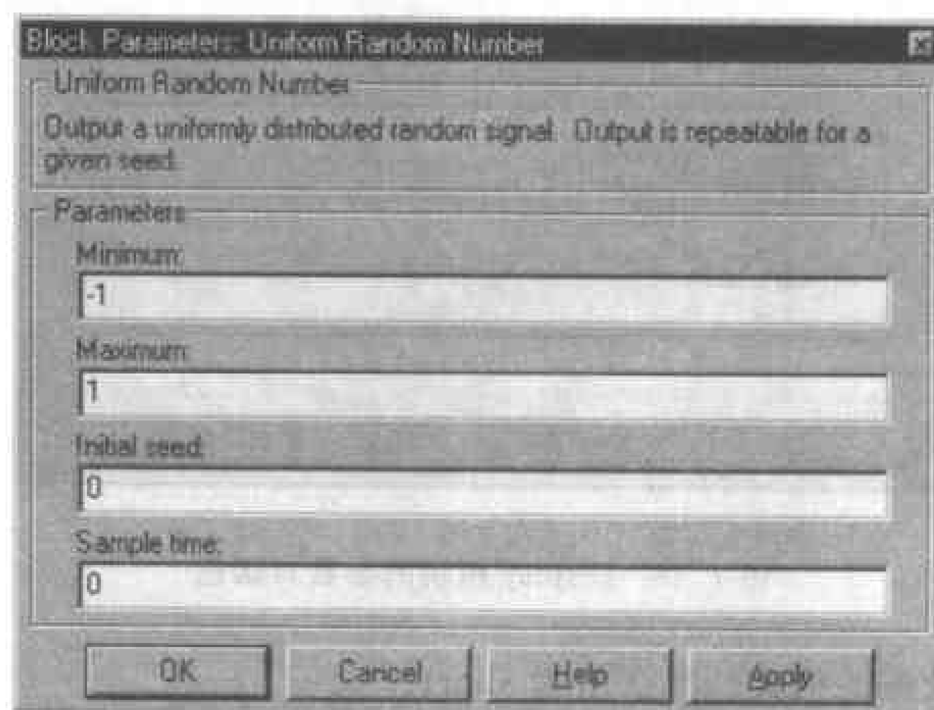


图 7.17　Uniform Random Number 模块的参数对话框

- Minimum：时间间隔的最小值。默认值为－1。
- Maximum：时间间隔的最大值。默认值为 1。
- Initial seed：随机序列发生器的起始种子。默认值为 0。
- Sample time：采样周期。默认值为 0。

3. 特性

- 数据类型：Uniform Random Number 模块输出双精度型实信号；
- 采样时间：连续、离散，或继承；
- 可矢量化。

7.5　接收器库模块

接收器模块包括显示和输出控制两类模块。

编者提示：在任何一个模型中，接收器模块和输入源模块是必不可少的，输入源模块和接收器模块是 Simulink 模型的最基本组成。换句话说，没有它们的模型就不是一个完整的模型。

7.5.1　Display 模块

1. 功能描述

显示模块输入的值。

2. 参数和对话框

Display 模块的参数对话框如图 7.18 所示。

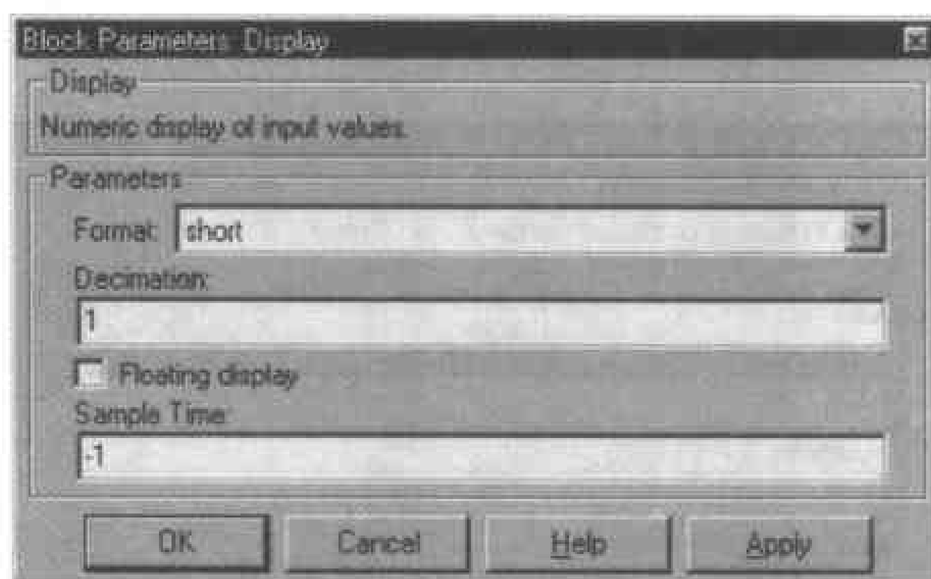


图 7.18　Display 模块的参数对话框

- Format：显示数据的格式。默认值为 short。
- Decimation：数据的显示进程。默认值为 1，显示每一个输入点。
- Floating display：若选中，模块的输入端口消失，模块成为浮点显示模块。
- Sample time：显示点数据的采样时间。

编者提示：

用户通过 Format 选项可以控制显示的格式：

- short，5 位定点；
- long，15 位定点；
- short_e，16 位浮点；
- bank，标准元和分（无 $ 符号或逗号）。

选中 Floating display 复选框后，模块将以浮点方式显示。模块的输入端口消失，模块显示信号的值。如果选中 Floating display，用户必须关闭 Simulink 缓冲器再用部件。欲知详情，请参看 5.3.3 节中有关“禁止优化 I/O 存储”的内容。

数据显示量和显示数据的时间由模块参数决定：

• 用户通过 Decimation 参数指定每第 n 次采样显示数据，其中 n 为抽样因子。默认值为 1，即每一时间步都显示数据。

• 用户通过 Sample time 参数可以指定用于显示点数据的采样时间间隔。当使用变步仿真器（时间步的时间间隔可能不同）时，就需要用到该参数。若值为 −1（默认），模块在显示数据时，将忽略采样时间间隔。

如果模块的输入是矢量，用户可以调整模块的尺寸以显示更多的矢量元素。可纵向也可横向调整，模块随之增加显示区域。黑三角形表明模块没有显示输入矢量的所有元素。例如，图 7.19 为把一个矢量传递给一个 Display 模块的模型。上面的模型为调整前的模块，注意黑三角形。下面的模型为调整后的模块，显示两个输入矢量元素。

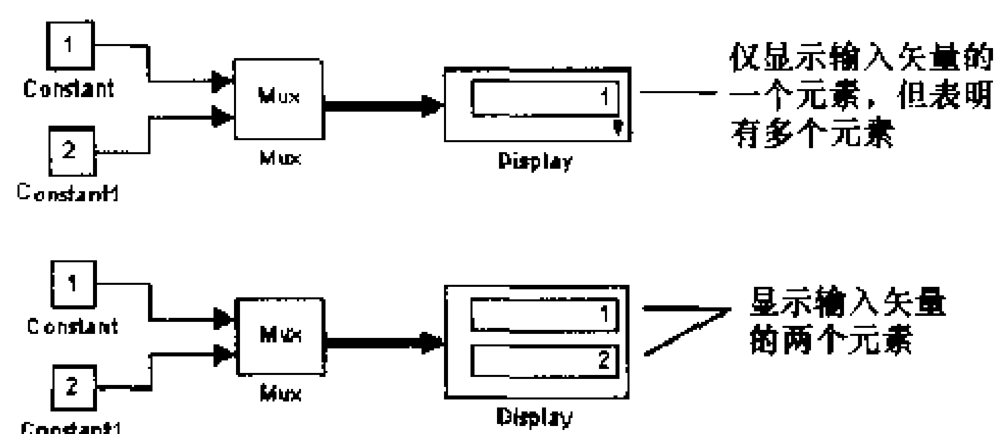


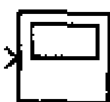
图 7.19 Display 模块应用举例

3. 特性

• 数据类型：输入和输出任意实或复信号；

• 采样时间：从驱动模块继承；

• 可矢量化。

7.5.2 Scope 模块 

1. 功能描述

Scope 模块显示仿真时间内的输入。Scope 模块可以有多个显示坐标（每个端口一个），所有坐标有共同的时间坐标和各自独立的 y 坐标。Scope 模块允许用户调整时间量和输入值的显示范围。用户可以移动和调整 Scope 模块窗口的尺寸，并且在仿真时间内修改 Scope 模块的参数。

当用户启动仿真时，Simulink 虽然把数据写入连接的 Scope 模块中，但并不打开 Scope 窗口。要想显示，则必须在仿真前或在仿真结束后双击模型中的 Scope 模块，就可打开。

若输入信号是连续的，Scope 产生一个连点波形图；若是离散的，Scope 就产生阶梯状波形图。Scope 模块提供多个用于放大显示数据的工具栏按钮，可以显示所有输入数据，保存前一个仿真时间的坐标设置，以用于下一个限定数据的显示以及将数据保存到工作空间。图 7.20 标注了 Scope 窗口上各工具栏按钮的作用。

(1) 显示矢量信号。在显示一个矢量信号时，Scope 根据下列次序使用不同的颜色：黄(yellow)、绛红(magenta)、青色(cyan)、红(red)、绿(green)以及深蓝(dark blue)。若信号显示

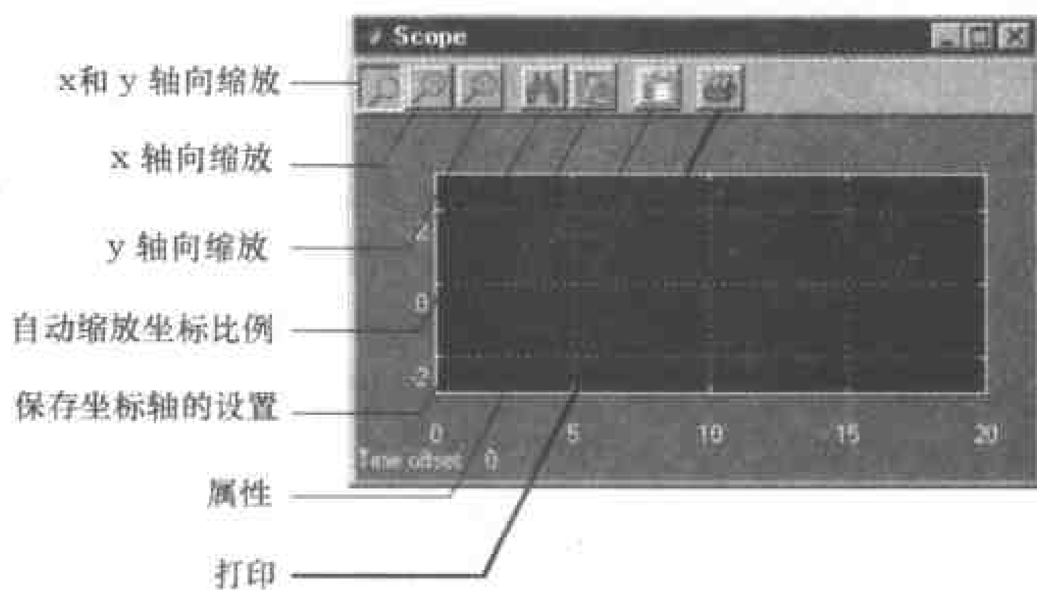


图7.20 Scope模块窗口及控件

数超过6个，Scope模块按上述次序循环使用。

(2) Y坐标限。用户在某个坐标上使用鼠标右键点击，然后选择"Properties.."就出现图7.21所示的对话框。

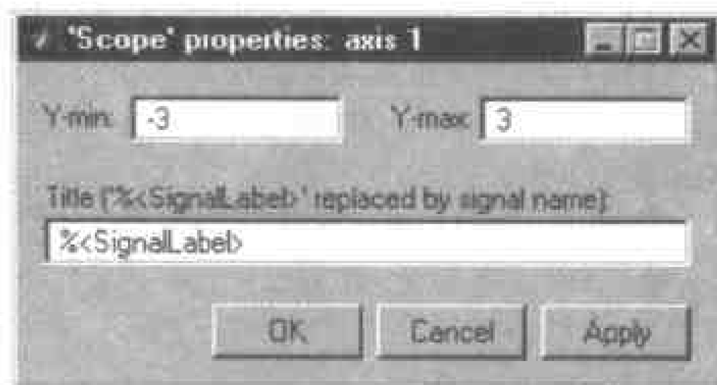


图7.21 Scope模块的属性对话框

- Y－min：输入y坐标的最小值。
- Y－max：输入y坐标的最大值。
- Title：输入图形的标题。用户可通过将输入信号标注%<SignalLabel>作为整个标题字符串的一部分把信号标注包含在标题中，实际操作时用输入信号标注代替%<SignalLabel>。

(3) 时间偏置。图7.22表明了Scope模块显示vdp模型的输出情况。仿真时间40s。

编者提示：此图仅显示仿真的最后20s。Time offset栏显示对应水平坐标0点的时间。因此，用户可以通过该参数给固定时间范围加一个偏置以获得明确的时间。

(4) 自动缩放显示坐标轴的比例。图7.23表明了按Auto-scale工具栏按钮后Vdp模型的输出。它自动确定两个坐标轴的比例以显示全部仿真数据。在此，y坐标轴不变，因为它已经设置成受限。

在运行仿真时若点击Auto-scale按钮，坐标轴将根据显示在当前屏幕上的数据自动调整比例，并且自动比例限制被保存为默认值。这使得用户可以将相同的限制用于其他仿真。

(5) 缩放。可以在同时放大X和Y轴上的数据，或分别放大。当仿真运行时，放大或缩小不起作用。

为在同时放大两个轴向上的数据，选择最左边的Zoom工具栏按钮。然后，使用约束框定义缩放区域(图7.24)。当用户释放鼠标后，Scope显示该区域的数据(图7.25)。用户也可以

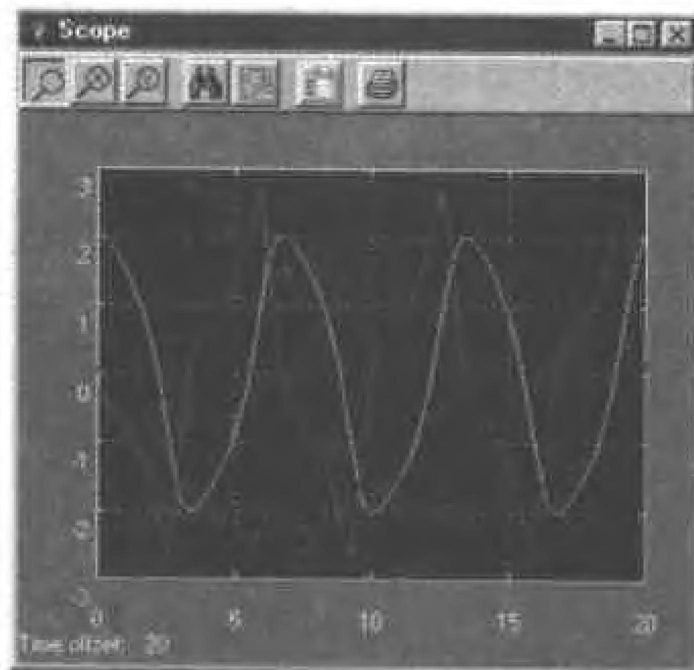


图 7.22 Scope 模块的属性对输出波形的影响

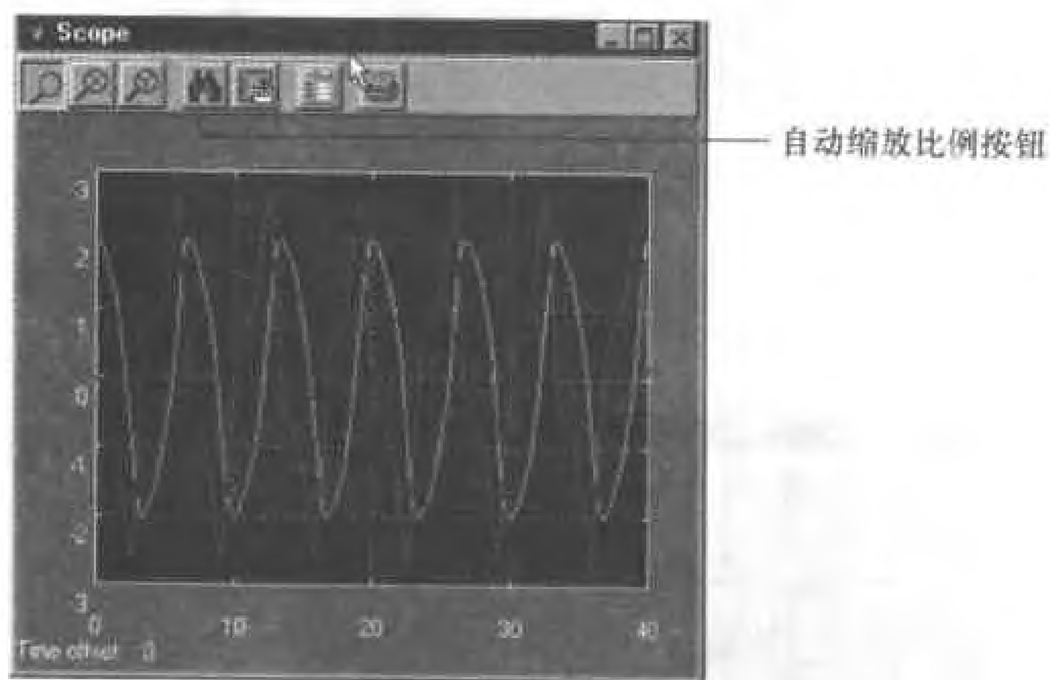


图 7.23 Scope 模块的比例属性对输出波形的影响

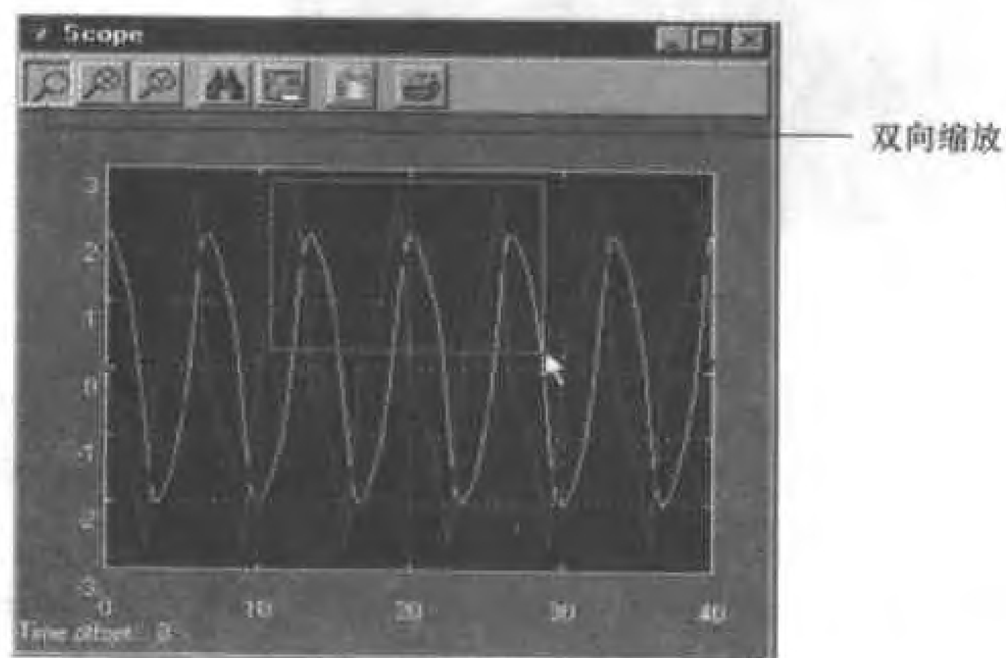


图 7.24 包在一个约束框内显示数据区

点击缩放区域的一个点。若示波器上有多个 Y 坐标，并且用户只对一个 x－y 坐标系进行缩放，则所有 x－y 坐标系的 X 限将会同时改变，这是因为所有坐标系必须共享相同的时间基准。

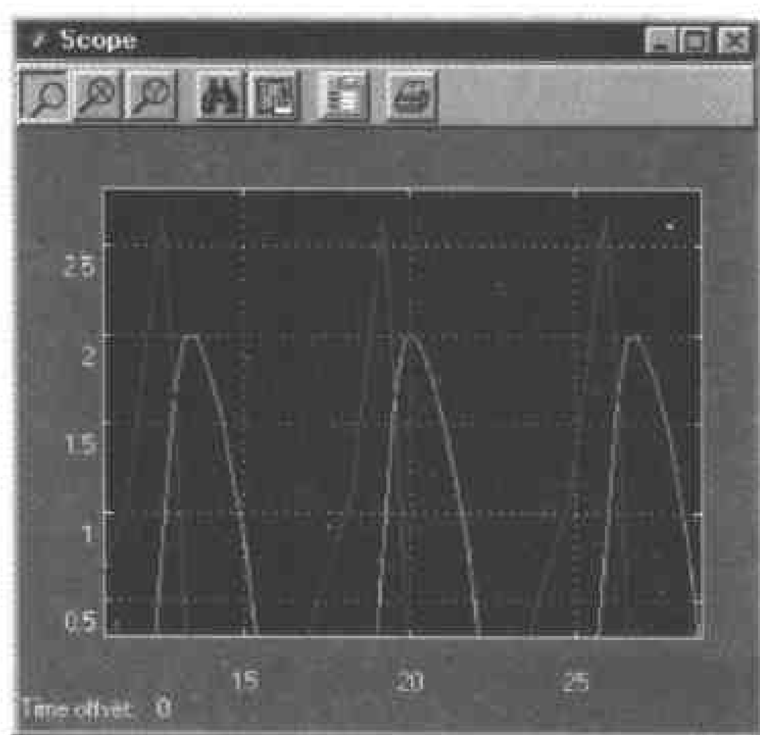


图 7.25　释放鼠标后出现的缩放区

点击中间一个 Zoom 工具栏按钮，仅放大 x 轴向的数据(图 7.26)。通过改变缩放区域头或尾部的指针(按下鼠标并保持，将指针移至另一端)定义缩放区域。

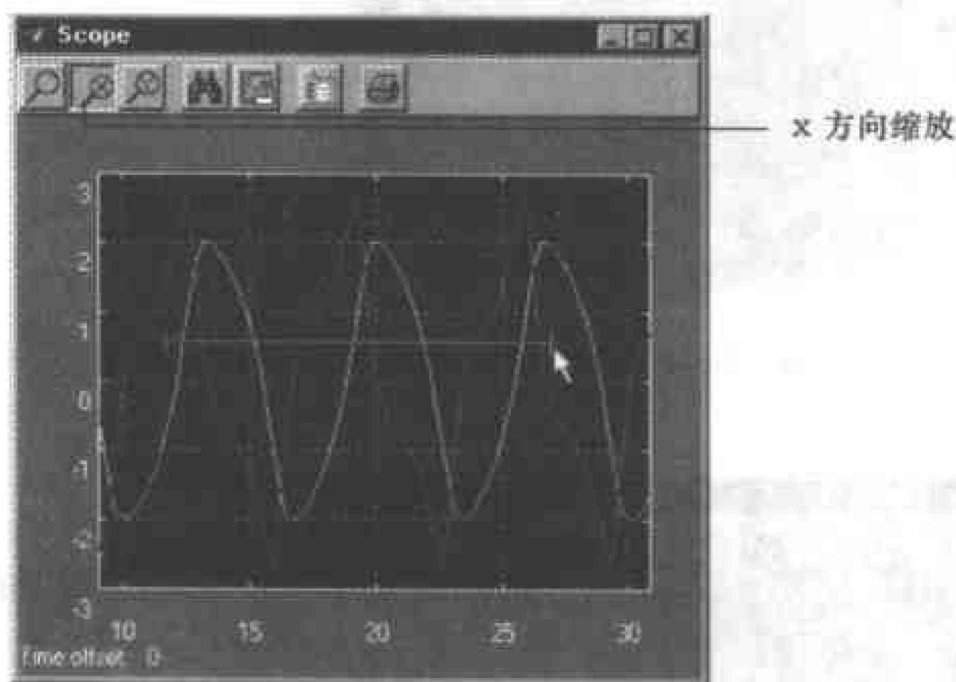


图 7.26　在定义缩放区域和释放鼠标之前的 Scope

当用户释放鼠标后，Scope 显示被放大的区域。用户也可以点击要放大的区域中的一点。

对 Y 轴向的操作基本相同，不同之处在于定义缩放区域之前，用户应按最右边的 Zoom 工具栏按钮。此外，用户也可以点击要放大的区域中的一点。

(6) 保存坐标轴设置(Save axes settings)。用户通过 Save axes settings 工具栏按钮(见图 7.27)可将当前 X 和 Y 坐标轴的设置储存起来以便用于下一次仿真。

保存应在放大显示数据区后做，这样可在另一个仿真中看见相同的区。时间范围则是从当前 X 轴限得到的。

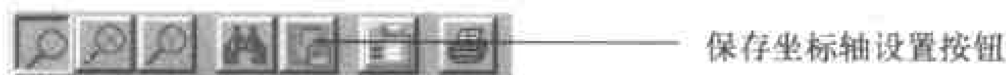


图 7.27 坐标轴设置按钮

(7) Scope 属性(Properties)。用户通过选择 Properties 工具栏按钮(见图 7.28)可以完成的设置项目有:改变坐标轴限、设置坐标轴数、时间范围、记号标签、采样参数以及保存选项。



图 7.28 坐标轴属性按钮

在点击 Properties 按钮后,会出现图 7.29 所示的对话框。

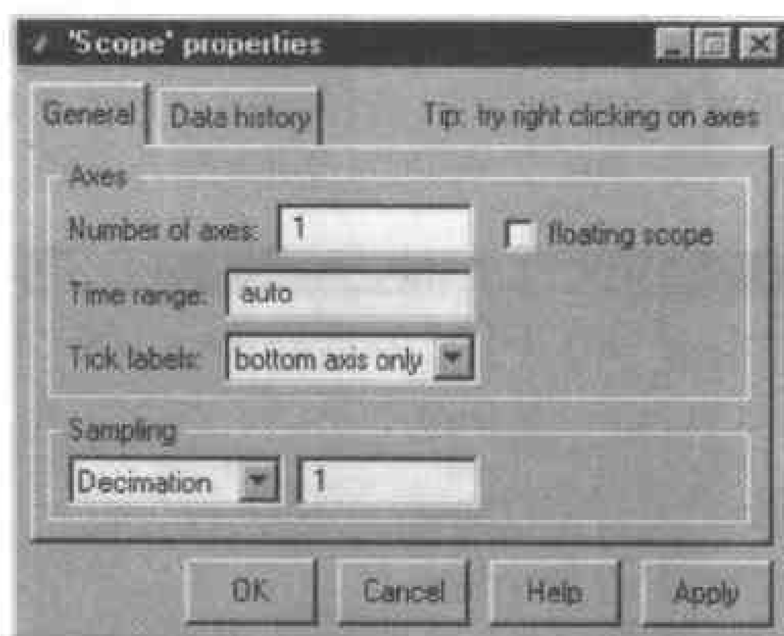


图 7.29 属性对话框

对话框有两个选项:General 和 Data history。

用户可以在 General 标签下设置坐标轴参数、时间范围和记号标签。也可以选择 floating scope 选项。

- Number of axes:在此栏中设置 Y 轴的数量。除了浮显(浮动显示)的示波器外,对 Scope 模块所能包含的坐标轴数没有限制。所有坐标轴共享时间基准(X 轴),但 Y 轴独立。注意:坐标轴数等于输入端口数。
- Time range:在本栏内输入一个数值(s)或 auto,可以改变 X 轴限。输入的数值使每个屏显示对应输入时间(s)的数据。输入 auto 可将 X 轴设置为仿真周期。不能在本栏中输入变量名。
- Tick labels:在本下拉菜单中,用户可选择在所有坐标轴、一个坐标轴或在底部坐标轴上标上标签。
- Floating scope:若用户要一个浮显示波器,可以选中 Floating scope 复选框。浮显示波器就是一个能显示单路或多路信号的示波器。

在一个模型中加入浮显示波器的方法:将一个 Scope 模块复制到模型窗口内,打开模块,

选择模块工具栏中的 Properties 按钮，然后选择 General 选项并选中 Floating scope 复选框。

在仿真过程中使用一个浮显示波器：先打开模块，选择一路信号显示，按下 shift 键＋点击另一路以便选择多路。有可能需要按下模块工具栏的 Auto-scale data 按钮以寻找信号，调整坐标轴。

编者提示：浮显示波器不能提供多坐标系。一个模型中可以有多个流显示波器，但即使有多个，因为它们显示的内容是一样的，所以实际上无用。在 Simulink 4.1 版本中已提供独立的 Floating scope 模块。

若用户计划在一个仿真过程中使用浮显示波器，则必须禁止缓冲器重复使用。参看第 5 章中的“禁止优化 I/O 存储器”。

• Sampling：在 Decimation 选择右边的数据栏内输入一个数值以指定抽取因子。选中并在右边的栏中输入数值，以显示一个采样周期内的数据。

(8) 数据采集和显示控制。用户可以通过 Data History 选项的设置栏控制 Scope 存储和显示的数据量(见图 7.30)。也可以在 Data History 标签中选择将数据存入工作空间。用户点击 Apply 或 OK 按钮应用当前参数和选项。栏内所显示的值就是在下一个仿真中应用的值。

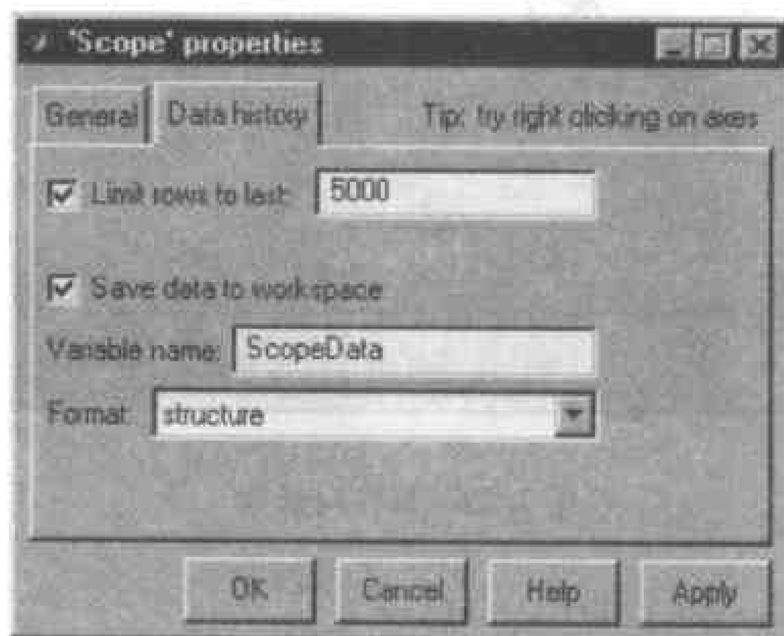


图 7.30　属性对话框(数据历史)

• Limit rows to last：用户可以通过 Limit rows to last 复选框并在数据栏内输入一个值限制行数。Scope 根据数据历史记录进行缩放和自动调整坐标轴比例。假设行数被限制为 1 000行而仿真过程产生了 2 000 行，则仅显示最后 1 000 行的数据。

• Save data to workspace：用户可以通过 Save data to workspace 复选框自动保存仿真结束所采集的数据。且用户选中此选项，则 Variable name 和 Format 栏就被激活。

• Variable name：在 Variable name 栏内输入一个变量名。指定的变量名在所有模型的数据登录变量中必须是惟一的。其他数据登录变量在 Scope 模块以及 To Workspace 模块上定义，仿真返回变量如时间、状态数及输出等。能将 Scope 数据存入工作空间意指没有必要将相同的数据流传递给 Scope 模块和 To Workspace 模块。

• Format：数据可以三种格式保存：Matrix，Structure 或 Structure with time。当 Scope 仅有一个坐标轴时用 Matrix。若 Scopes 有多于一个坐标轴，且用户不要存储时间数据，使用

Structure；若需存储时间数据则使用 Structure with time。

(9) 打印一个 Scope 窗口的内容。单击 Print 图标(在 Scope 工具栏最右边，见图 7.31)，打开 Scope Properties 对话框。



图 7.31 打印图标按钮

2. 特性

- 数据类型：Scope 模块接受任意类型的实信号，包括纯矢量；
- 采样时间：从驱动模块继承或设置；
- 状态数：0。

7.5.3 Stop Simulation 模块

1. 功能描述

当输入为非 0 时，Stop Simulation 模块停止仿真。在仿真停止之前，完成当前时间步内的仿真。若模块输入是矢量，则任何一个非 0 元素都将使仿真停止。

用户可以将本模块与 Relational Operator 模块联合使用去控制停止仿真的时间。例如，如图 7.32 的模型在输入信号达 10 时，停止仿真。

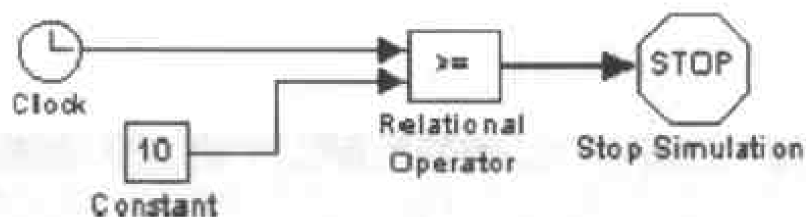


图 7.32 模块应用举例

2. 对话框

Stop Simulation 模块的参数对话框如图 7.33 所示。

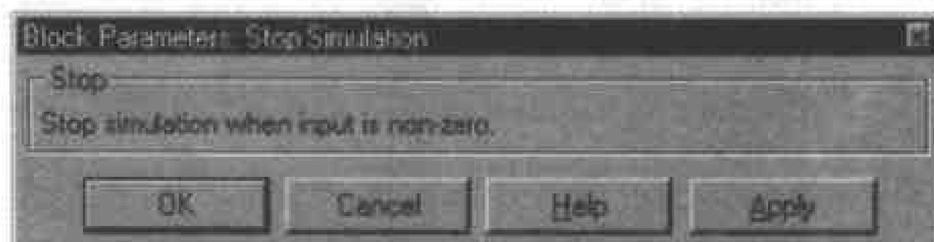


图 7.33 Stop Simulation 模块的参数对话框

3. 特性

- 数据类型：Stop Simulation 模块接受双精度型实信号；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.5.4　To File 模块 untitled.mat

1. 功能描述

To File 模块将其输入写入 MAT-file 文件内的一个矩阵中。模块在一个时间步内写一列，第一行为仿真时间，其余为输入数据，对应输入矢量元素。矩阵的形式为：

$$\begin{bmatrix} t_1 & t_2 & \cdots & t_{\text{final}} \\ u1_1 & u1_2 & \cdots & u1_{\text{final}} \\ \vdots & \vdots & \vdots & \vdots \\ un_1 & un_2 & \cdots & un_{\text{final}} \end{bmatrix}$$

From File 模块不需任何更改就可以使用由 To File 模块写的数据。特别要注意，From Workspace 模块所读数据的矩阵形式是 To File 模块写的数据矩阵的转置阵。

在仿真结束后，模块才写数据和仿真时间。模块图标显示所指定的输出文件名。

写入的数据量和写入数据的所需时间步由模块参数决定：

- Decimation 参数允许用户写入每第 n 个采样数据，其中 n 为抽取因子。
- Sample time 参数允许用户指定一个收集数据的采样时间。在使用一个变步仿真器(时间步的间隔可以不同)时，要用到此参数。默认值 -1 使模块在决定要写入那一个数据时继承其驱动模块的采样时间。

若在仿真开始时，文件已经存在，则文件内容将被覆盖。

2. 参数和对话框

To File 模块的参数对话框如图 7.34 所示。

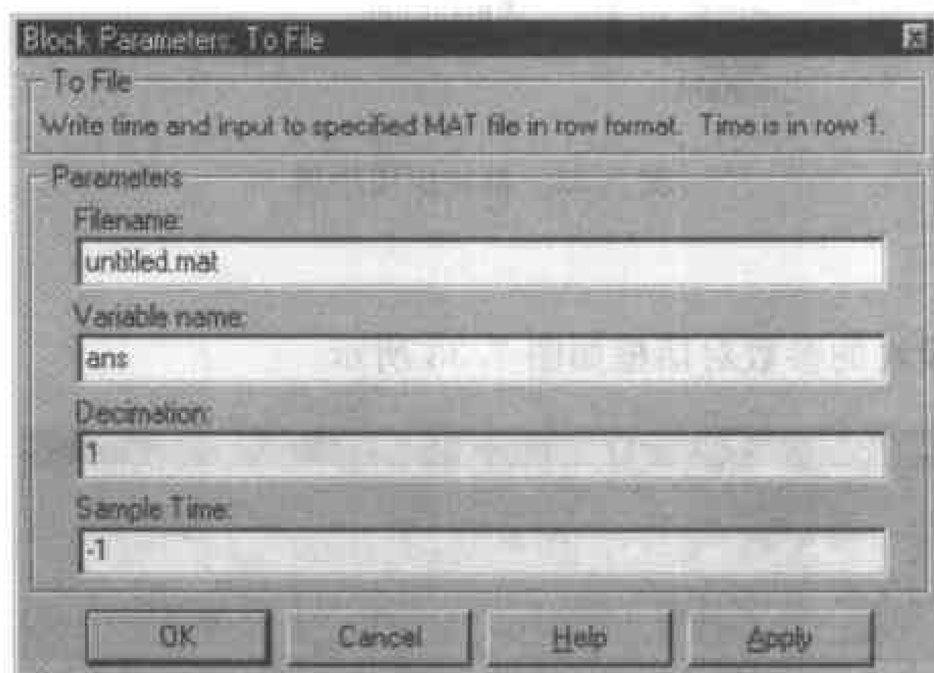


图 7.34　To File 模块的参数对话框

- Filename：保存矩阵的 MAT-file 文件名。
- Variable name：包含在命名文件中的矩阵名。
- Decimation：抽取因子。默认值为 1。
- Sample time：收集数据的采样时间。

3. 特性

- 数据类型：To File 模块接受双精度型实信号；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.5.5 To Workspace 模块 simout

1. 功能描述

To Workspace 模块将其输入写入工作空间。模块将其输出写入到一个由模块 Variable name 参数命名的矩阵或结构中。Save format 参数决定输出的格式：

(1) 矩阵格式(Matrix)。矩阵的形式：

$$\begin{bmatrix} u1_1 & u2_1 & \cdots & un_1 \\ u1_2 & u2_2 & \cdots & un_2 \\ \vdots & \vdots & \vdots & \vdots \\ u1_{final} & u2_{final} & \cdots & un_{final} \end{bmatrix}$$

写入的数据量和写入数据的所需时间步由模块参数决定：

- Maximum number of rows 参数表示保存的数据行数。若仿真产生的行数多于指定的最大值，将只保存最新产生的数据，欲捕获所有数据，设置参数为 inf。
- Sample time 参数允许用户指定一个收集数据的采样时间。在使用一个变步仿真器(时间步的间隔可以不同)时，要用到此参数。默认值 −1 使模块在决定要写入那一个数据时继承其驱动模块的采样时间。

在仿真过程中，模块将数据写入一个内部缓冲器中。当仿真结束或暂停时，这些数据被写入工作空间。模块图标形式写入数据的矩阵名。

(2) 结构格式(Structure)。由三个字段结构组成：时间、信号和模块名。时间字段是空的。模块名字段包含了 To Workspace 模块名。信号字段包含了信号值矩阵。

(3) 时间结构格式(Structure with Time)。与 Structure 格式类同，但时间字段包含了一个仿真时间步矢量。

(4) 用 From File 模块使用保存过的数据。用 To Workspace 模块写的数据若要保存并且过后要用 From File 模块读取，则数据中必须加入时间，而且矩阵必须转置。请参看“From 模块”。

(5) 用 From Workspace 模块使用保存过的数据。用 To Workspace 模块写的数据若要在另一个仿真中使用 From Workspace 模块时再次使用，则数据必须包括仿真时间值。输入时间量的方法取决于保存格式。

编者提示：若保存格式为结构格式，用户可以通过选择 Structure with Time 为 Save format 值输入时间量。模块将仿真时间以矢量形式放在输出结构的 time 成分中。若保存格式是矩阵格式，则必须把一列仿真时间加入到矩阵中。有两种加入方法：

- 利用 Clock 模块的多路输出作为 To Workspace 矢量输入行的第一个元素。
- 在 Simulation Parameters 对话框中或在命令行指定时间为一个返回值(参看第 4 章)。

当仿真结束后，用户可以用下列命令将时间矢量(t)与矩阵进行链接：

```
matrix = [t; matrix];
```

2. 举例说明

例 1. 开始时间为 0，Maximum number of rows 为 100，Decimation 为 1，Sample time 为 0.5。To Workspace 模块在时间 0，0.5，1.0，1.5，…秒采集了 100 个最大值点。指定 Decimation 为 1，命令模块在每一个时间步写数据。

例 2. Maximum number of rows 为 100，Sample time 为 0.5，但 Decimation 为 5。在该例中，模块在时间 0，2.5，5.0，7.5，…秒采集了 100 个点。指定的 Decimation=5 命令模块在每第五个采样写数据。采样时间保证了在这些点上写数据。

例 3. 除 Maximum number of rows 为 3 外，其他参数同第一个例子。在此情况下，仅最后三行采集的数据被写入工作空间。若仿真结束时间为 100，数据为对应于时间 99.0，99.5，和 100.0 s(3 个点)采集的数据。

3. 参数和对话框

To Workspace 模块的参数对话框如图 7.35 所示。

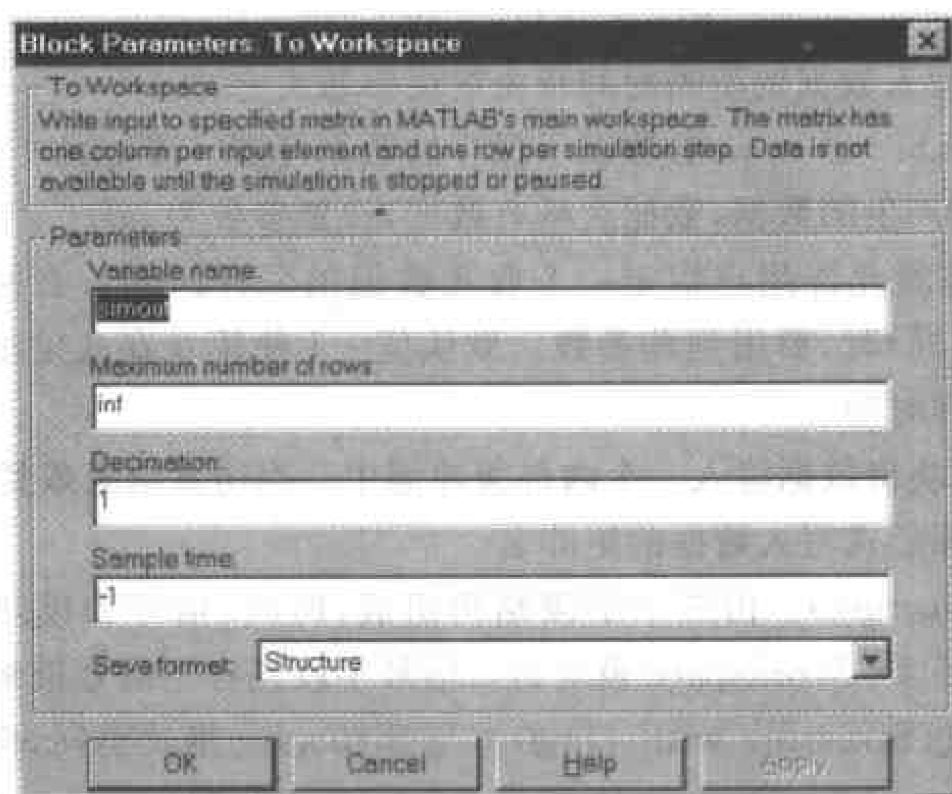


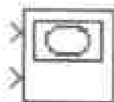
图 7.35 To Workspace 模块的参数对话框

- Variable name：数据矩阵名。
- Maximum number of rows：保存的最大行数(每一个时间步一行)。默认值为 1 000 行。
- Decimation：抽取因子。默认值为 1。
- Sample time：采集点的采样时间。
- Save format：将仿真输出保存到工作空间的格式。默认值为 structure 格式。

4. 特性

- 数据类型：To Workspace 模块可以将任意数据类型的实或复数据输入存入 MATLAB 工作空间；
- 采样时间：继承；

- 可矢量化。

7.5.6 XY Graph 模块 

1. 功能描述

XY Graph 模块将其输入的 X－Y 平面图显示在一个 MATLAB 图形窗口上。

本模块有两个输入。模块根据第一个输入的数据(X 方向)对第二个输入的数据(Y 方向)绘图。本模块在检查有限循环和其他双态数据等方面是很有用的。超出指定范围的数据将不显示。

Simulink 在仿真开始后为 XY Graph 模型中的模块打开一个图形窗口。

在命令窗口输入 lorenzs，可看到 XY Graph 模块的使用说明。

2. 参数和对话框

XY Scope 模块的参数对话框如图 7.36 所示。

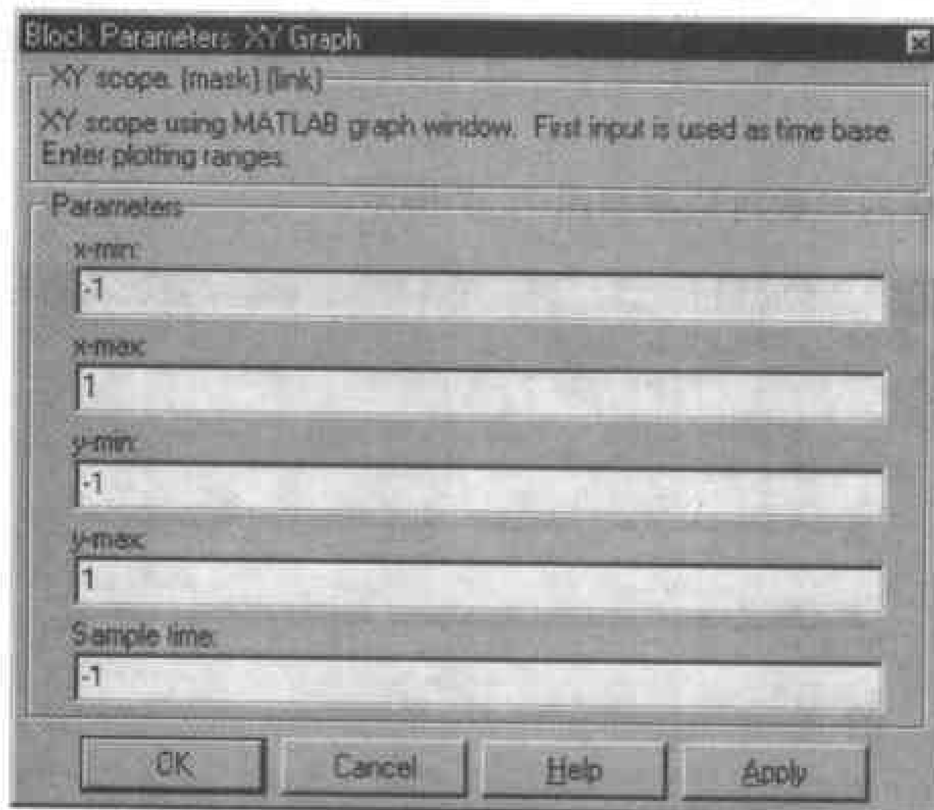


图 7.36 XY scope 模块的参数对话框

- x—min：x 轴最小值。默认值为—1。
- x—max：x 轴最大值。默认值为 1。
- y—min：y 轴最小值。默认值为—1。
- y—max：y 轴最大值。默认值为 1。
- Sample time：采样的时间间隔。默认值为—1，指采样时间由驱动模块决定。

3. 特性

- 数据类型：XY Graph 模块接受双精度型实信号；
- 采样时间：从驱动模块继承；
- 状态数：0。

7.6　离散系统库模块

由描述离散时间系统的模块组成。

7.6.1　Discrete Filter 模块

1. 功能描述

Discrete Filter 模块实现无限冲击响应(IIR)和有限冲击响应(FIR)滤波器。用户可用 Numerator 和 Denominator 参数指定以 z^{-1} 的升幂为矢量的分子和分母多项式的系数。分母的阶数必须大于或等于分子的阶数。参看“Discrete Transfer Fcn 模块”。

Discrete Filter 模块提供信号处理用 z^{-1}（延迟算子）多项式描述数字滤波器的方法。Discrete Transfer Fcn 模块提供自动控制用 z 描述离散系统的方法。当分子和分母的长度相同时，这两种方法是一样的。一个 n 元素的矢量描述 $n-1$ 次的多项式。模块图标根据分子和分母的定义不同显示不同结果。关于 Simulink 如何显示图标，请参看“Transfer Fcn 模块”的内容。

2. 参数和对话框

Discrete Filter 模块的参数对话框如图 7.37 所示。

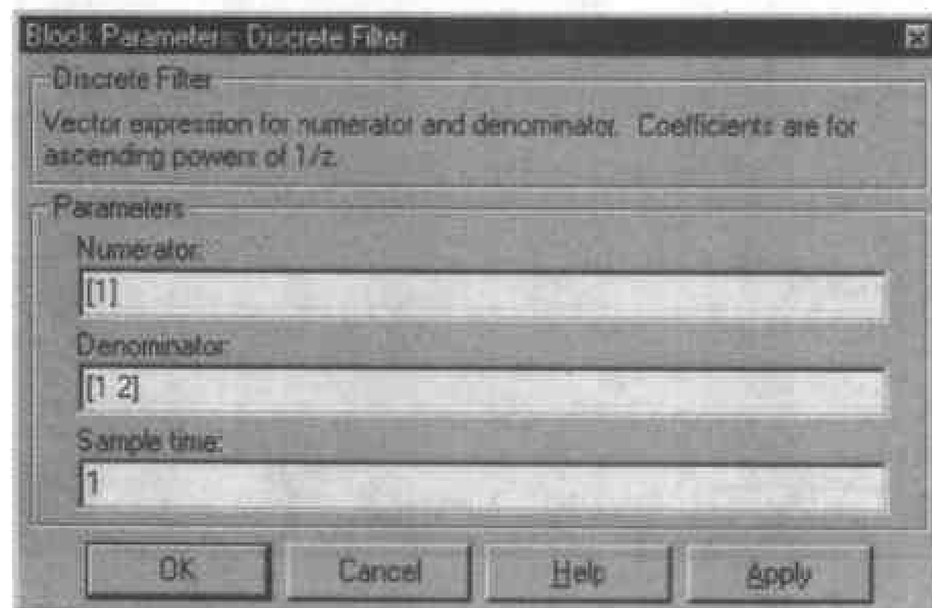


图 7.37　Discrete Filter 模块的参数对话框

- Numerator：分子系数矢量。默认值为[1]。
- Denominator：分母系数矢量。默认值为[1 2]。
- Sample time：两次采样的时间间隔。

3. 特性

- 数据类型：Discrete Filter 模块输入和输出双精度型实信号；
- 直通输出：仅当 Numerator 和 Denominator 参数的长度相等时；
- 采样时间：离散；
- 状态数：Denominator 参数－1；

7.6.2　Discrete State-Space 模块

y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)

1. 功能描述

Discrete State-Space 模块实现如下一个离散系统：

$$x(n+1)=Ax(n)+Bu(n)$$
$$y(n)=Cx(n)+Du(n)$$

其中：u 为输入；x 为状态；y 为输出。但矩阵系数必须满足以下要求（参见图 7.38）：

- A 必须是 $n\times n$ 矩阵，其中 n 为状态数量；
- B 必须是 $n\times m$ 矩阵，其中 m 为输入数量；
- C 必须是 $r\times n$ 矩阵，其中 r 为输出数量；
- D 必须是 $r\times m$ 矩阵。

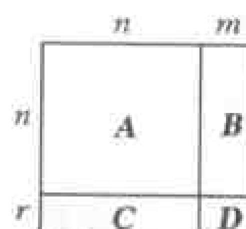


图 7.38　对矩阵系数的要求

模块接受一个输入，并产生一个输出。输入矢量宽度由矩阵 B 和 D 的列数决定。输出矢量宽度由矩阵 C 和 D 的行数决定。Simulink 将一个包含 0 的矩阵转换成一个利于相乘的稀疏矩阵。

2. 参数和对话框

Discrete State-Space 模块的参数对话框如图 7.39 所示。

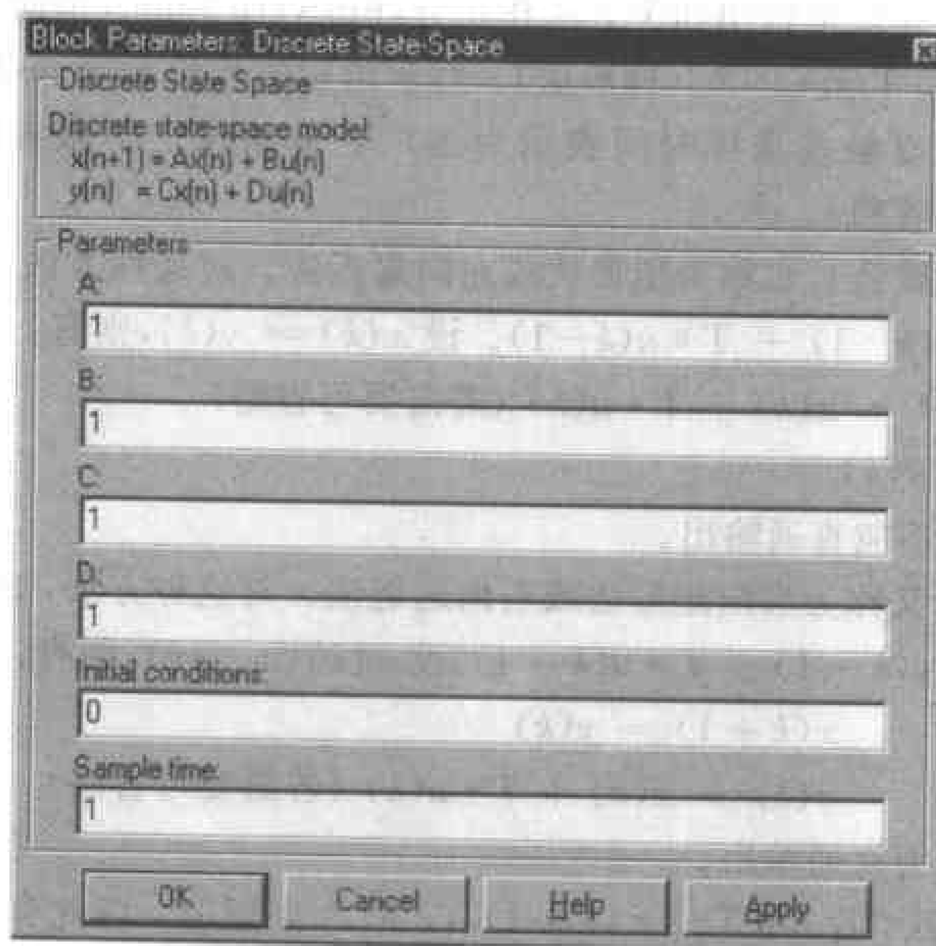


图 7.39　Discrete State-Space 模块的参数对话框

- A，B，C，D：在上述方程中定义的矩阵系数。
- Initial conditions：初始状态矢量。默认值为 0。
- Sample time：采样时间间隔。

3. 特性

- 数据类型：Discrete State Space 模块输入和输出双精度实信号；
- 直通输出：仅当 $D \neq 0$；
- 采样时间：离散；
- 标量扩展：初始条件；
- 状态数：取决于 A 的大小；
- 可矢量化。

7.6.3 Discrete-Time Integrator 模块

1. 功能描述

当用于构建纯离散系统时，Discrete - Time Integrator 模块可代替 Integrator 模块之用。Discrete - Time Integrator 模块允许用户完成下列任务：

- 在模块的对话框中定义初始条件或模块的输入；
- 输出模块状态；
- 定义积分上下限；
- 根据附加置位输入重新设置状态。

这些特点描述如下。

(1) 积分方法。该模块可以用如下方法作积分：Forward Euler(前向 Euler 法)，Backward Euler(后向 Euler 法)和 Trapezoidal(梯形法)。对于给定的步长 k，Simulink 更新 $y(k)$ 和 $x(k+1)$。T 是采样周期(以触发采样时间表示为 ΔT)。值根据上下限省略。在所有情况下，$x(0)=IC$(如果需要可省略)。

• 前向 Euler 法(默认)，也称为矩形法或左向逼近法。对这种方法，$1/s$ 用 $T(z-1)$ 来近似。因此有 $y(k) = y(k-1) + T*u(k-1)$。设 $x(k) = y(k)$，则有：

$$x(k+1) = x(k) + T*u(k) \text{ (若需要可省略)}$$
$$y(k) = x(k)$$

采用此法，输入端口 1 不能直通输出。

• 后向 Euler 法，又称为后向矩形法或右向逼近法。对这种方法，$1/s$ 用 $T*z(z-1)$ 来近似。因此有 $y(k) = y(k-1) + T*u(k-1)$。设 $x(k) = y(k)$，则有：

$$x(k+1) = y(k)$$
$$y(k) = x(k) + T*u(k) \text{ (若需要可省略)}$$

采用此法，输入端口 1 为直通输出。

• 梯形法。对此法，$1/s$ 用 $T/2*(z+1)/(z-1)$ 来近似。因此有：

$$y(k) = y(k-1) + T/2*(u(k) + u(k-1))$$

当 T 为固定(等于采样周期)时，设

$$x(k) = y(k-1) + T/2*u(k-1)\text{，则有：}$$
$$x(k+1) = y(k) + T/2*u(k) \text{ (若需要可省略)}$$
$$y(k) = x(k) + T/2*u(k) \text{ (若需要可省略)}$$

这里，$x(k+1)$ 是下一个输出的最佳估计。在 $x(k)! = y(k)$ 的情况下，它不是完备的状态。

当 T 是变量（从触发时间获得）时，有：

$$x(k+1) = y(k)$$

$$y(k) = x(k) + T/2 * (u(k) + u(k-1))\text{（若需要可省略）}$$

采用此法，输入端口 1 为直通输出。

该模块的图标反映出已选择的积分方法，如图 7.40 所示。

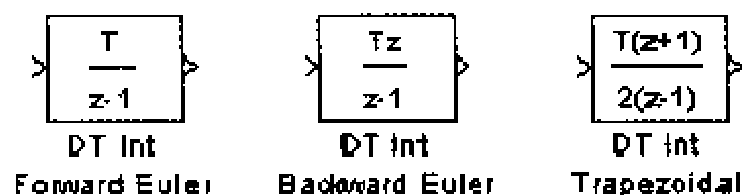


图 7.40　图标随积分方法不同而变

(2) 定义初始条件。用户可在模块的对话框中将初始条件定义为参数，或从一个外信号输入初始条件。

- 为将初始条件定义成模块的参数，须指定 Initial condition source 为 internal，并在 Initial condition 参数栏内输入值。

- 为了从一个外输入源调用初始条件，须指定 Initial condition source 为 external。在输入模块下，出现一个额外的输入端口，如图 7.41 所示：

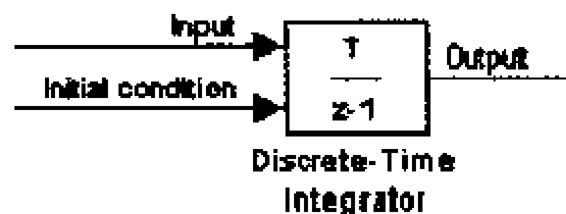


图7.41　在"external"时，模块端口的变化

(3) 使用状态端口。在下列两种情况下，用户必须用状态端口代替输出端口：

- 当模块的输出通过置位端口反馈回模块或馈入初始条件端口而产生代数循环时。对此种情况，见 bounce 模型。

- 当用户将状态从一个条件执行子系统传递到另一个时，有可能引起定时问题。详情请参见 clutch 模型。通过状态端口而不是输出端口传递状态可以解决该问题。即使数值是相同的，Simulink 也会以稍微不同的时间产生之，这样可使模型免于上述问题的影响。用户可通过选择 Show state port 复选框输出模块的状态。

在默认状态，状态端口出现在模块的顶部，如图 7.42 所示。

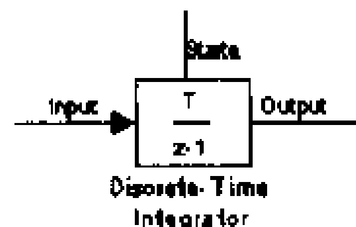


图 7.42　模块端口示意

(4) 设置积分限。为避免输出超出所指定的电平，须选择 Limit output 复选框并在相应的参数栏中填入积分限。这样，模块就以定积分形式工作。当输出落在积分限以外，将暂停积分以防止积分运算的中止。在仿真期间，用户可以改变积分限，但不能改变积分的性质。输出由以下情况决定：

- 当积分小于 Lower saturation limit（饱和极限）并且输入为负时，输出将维持在 Lower saturation limit；

• 当积分落在 Lower saturation limit 和 Upper saturation limit 之中，积分输出；

• 当积分大于 Upper saturation limit 并且输入为正时，输出维持在 Upper saturation limit。

为了产生显示状态正在限定的信号，选中 Show saturation port 复选框。一个饱和（saturation port）端口就会出现在模块的下方，如图 7.43 所示。

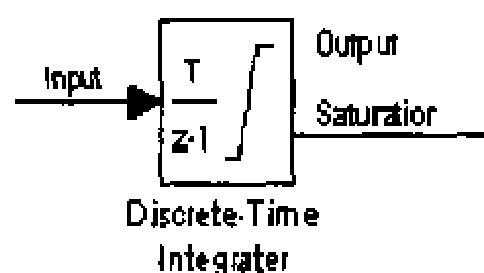


图 7.43　模块的饱和端口示意

信号为以下三种之一：

• 1 表示上极限正在使用；
• 0 表示积分是不定积分；
• −1 表示下极限正在使用。

当 Limit output 选项被选中后，模块有三个过零检测：一个探测模块到达饱和上限的时间；一个探测模块到达饱和下限的时间；一个探测模块脱离饱和的时间。

(5) 状态复位。模块能将它的状态重新设置为基于外信号的初始条件。为了使模块重新设置状态，须选择 External reset 选项之一，触发器端口出现在模块输入端口的下方，并显示触发器类型，如图 7.44 所示。

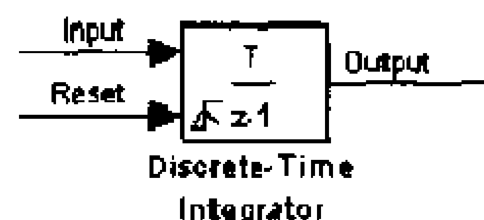


图 7.44　模块的触发端口示意

• 选中 rising 用于当置位信号处于上升沿时触发状态设置。
• 选中 falling 用于当置位信号处于下降沿时触发状态设置。
• 选中 either 用于当置位信号处于上升沿或下降沿时触发状态设置。

剩下的端口为直通输出。如果模块的输出直接或通过一系列直通输出模块被反馈到该端口，就导致代数循环。将模块状态馈送到置位端口可以解决这个问题。选中 Show state port 复选框用以访问模块的状态。

(6) 选中所有选项。当所有选项都被选中时，模块图标看起来如图 7.45 所示。

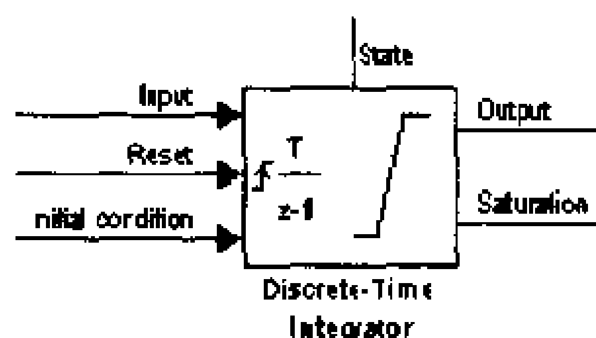


图 7.45　模块图标全图

2. 参数和对话框

Discrete - Time Integrator 模块的对话框如图 7.46 所示。

• Integrator method：积分法。默认值为 Forward Euler。

• External reset：当在置位信号中出现一个触发器事件（rising，falling 或 either）时，将模块的状态设置为初始条件。

• Initial condition source：从 Initial condition 参数（如果设置为 Initial）或从一个外部模块（如果设置为 external）获取模块的状态初始条件。

• Initial condition：状态初始条件。设置 Initial condition source 参数为 Initial。

• Limit output：若选中，状态值将被限制在 Lower saturation limit 和 Upper saturation limit 参数之间。

• Upper saturation limit：积分上限。默认值为 inf。

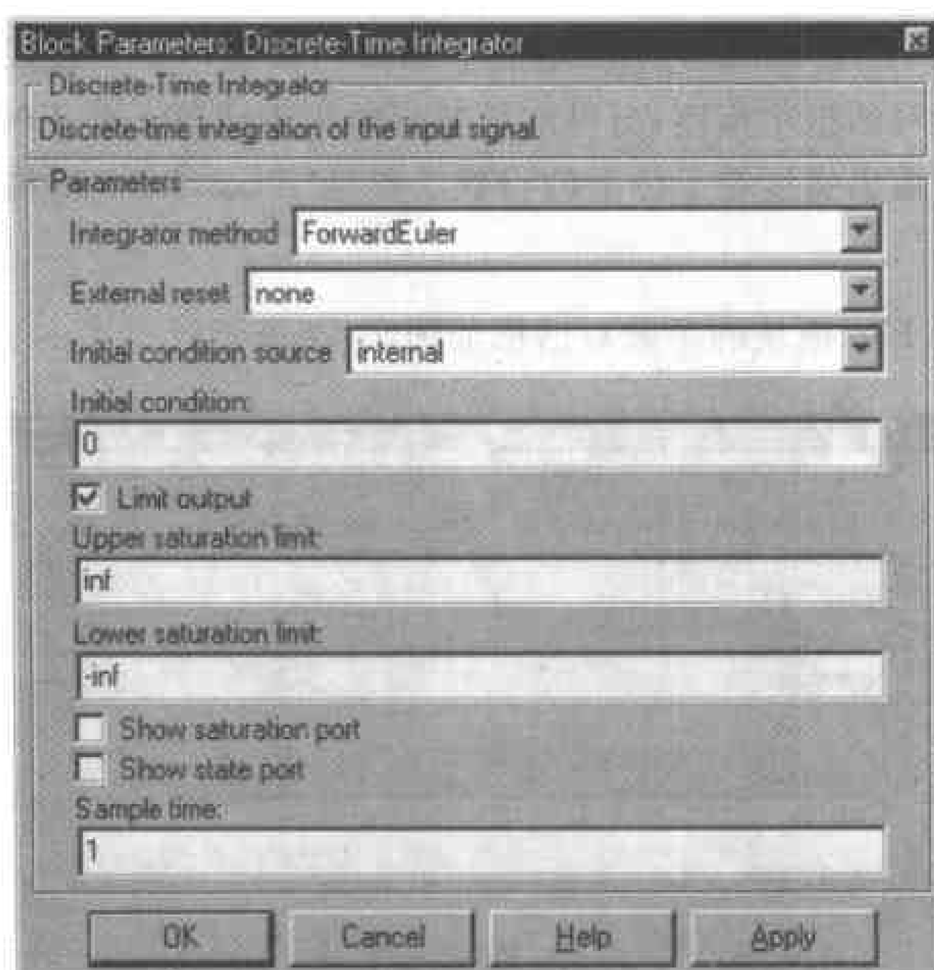


图 7.46 Discrete-Time Integrator 模块的对话框

- Lower saturation limit：积分下限。默认值为－inf。
- Show saturation port：若选中，给模块加一个饱和端口。
- Show state port：若选中，给模块加一个模块状态的输出端口。
- Sample time：采样之间的时间间隔。默认值为 1。

3. 特性

- 数据类型：Discrete-Time Integrator 模块接受并输出双精度型实信号；
- 直通输出：置位和外部初始条件端口；
- 采样时间：离散；
- 标量扩展：参数；
- 状态数：从驱动模块和参数继承；
- 可矢量化；
- 过零检测：一个探测置位，一个探测上和下极限，一个探测脱离饱和的时间。

7.6.4 Discrete Transfer Fcn 模块



1. 功能描述

Discrete Transfer Fcn 模块实现由下列方程描述的 z 变换传递函数：

$$H(z)=\frac{num(z)}{den(z)}=\frac{num_0 z^n+num_1 z^{n-1}+\cdots+num_m z^{n-m}}{den_0 z^n+den_1 z^{n-1}+\cdots+den_n}$$

其中：$m+1$ 和 $n+1$ 分别为分子和分母多项式系数的个数，***num*** 和 ***den*** 包含有分子和分母两个多项式的系数，并以 z 的递减次幂的形式排列。***num*** 可以是一个矢量或矩阵，***den*** 则必须是矢量，两者均须在模块的对话框中指定。分母的阶数必须大于或等于分子的阶数。

该模块输入矢量，输出宽度等于分子的行数。参见“Transfer Fcn 模块”。

2. 参数和对话框

Discrete Transfer Fcn 模块的参数对话框如图 7.47 所示。

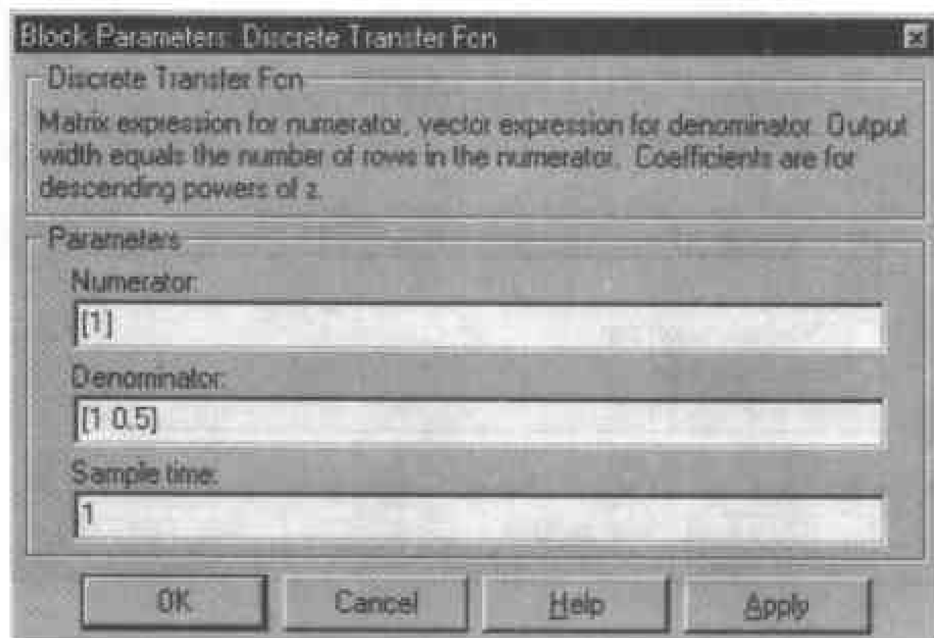


图 7.47　Discrete Transfer Fcn 模块的参数对话框

- Numerator：分子多项式系数矢量行。可以指定含有多行的矩阵产生多行输出。默认值为[1]。
- Denominator：分母多项式系数矢量行。默认值为[1 0.5]。
- Sample time：两次采样之间的时间间隔。默认值为 1。

3. 特性

- 数据类型：Discrete Transfer Function 模块输入和输出双精度实信号；
- 直通输出：仅当 Numerator 和 Denominator 长度相等时；
- 采样时间：离散；
- 状态数：Denominator 长度－1。

7.6.5　Discrete Zero-Pole 模块

1. 功能描述

Discrete Zero-Pole 模块实现一个以指定的零点、极点、增益及延迟算子 z 表示的离散系统。一个传递函数可以用零点－极点－增益的形式表示。对于一个 MATLAB 下的单输入、单输出系统，传递函数为：

$$H(z)=K\frac{Z(z)}{P(z)}=K\frac{(z-Z_1)(z-Z_2)\cdots(z-Z_m)}{(z-P_1)(z-P_2)\cdots(z-P_n)}$$

其中，$\boldsymbol{Z}$ 表示为零点矢量；$\boldsymbol{P}$ 为极点矢量；$\boldsymbol{K}$ 为增益。极点数必须大于或等于零点数($n\geq m$)。若极点和零点是复数，它们必须是复共轭对。

模块的输入和输出宽度等于零点矩阵的行数。模块根据参数的指定形式在其图标内显示传递函数。参看“Zero-Pole 模块”。

2. 参数和对话框

Discrete Zero-Pole 模块的参数对话框如图 7.48 所示。

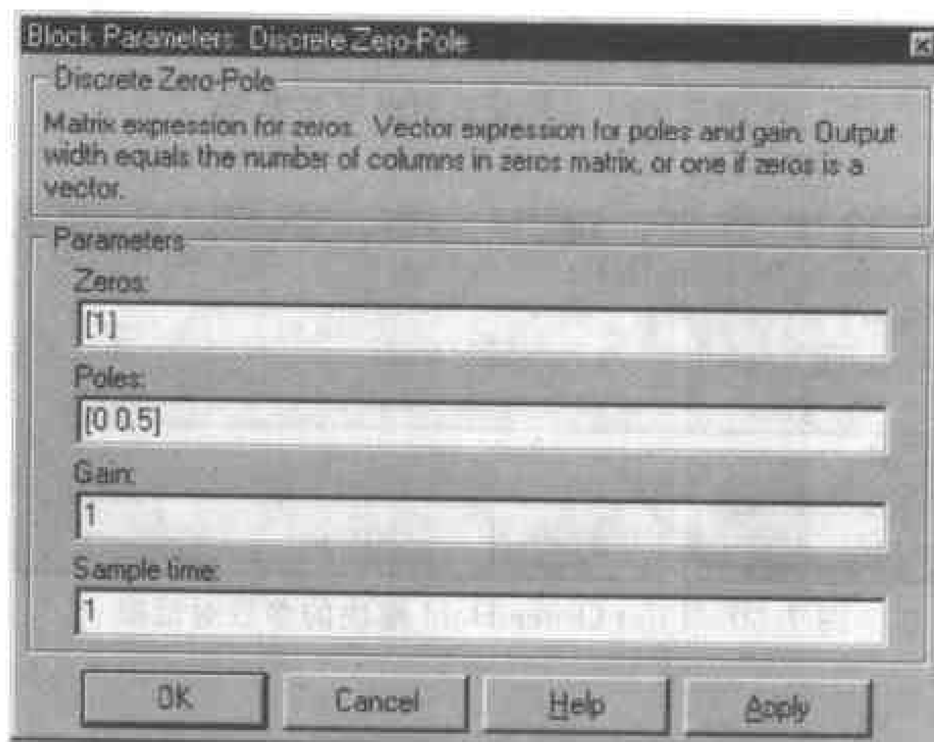


图 7.48 Discrete Zero-Pole 模块的参数对话框

- Zeros:零点矩阵。默认值是[1]。
- Poles :极点矢量。默认值是[0 0.5]。
- Gain:增益。默认值为 1。
- Sample time:采样时间间隔。

3. 特性

- 数据类型:Discrete Zero-Pole 模块接受和输出双精度型实信号;
- 直通输出:仅当零点和极点数相等;
- 采样时间:离散;
- 状态数:Poles 矢量的长度。

7.6.6 First-Order Hold 模块

1. 功能描述

本模块在指定的时间间隔实现一阶采样保持。该模块主要用于理论研究。

我们可以通过 demo 演示程序中的 fohdemo 来观察和比较零阶保持(Zero-Order Hold)与一阶保持(First-Order Hold)之间的不同。图 7.49 为 Sine Wave 模块的输出和经过 First-Order Hold 模块的输出比较。

2. 参数和对话框

First-Order Hold 模块的参数对话框如图 7.50 所示。

- Sample time:采样之间的时间间隔。

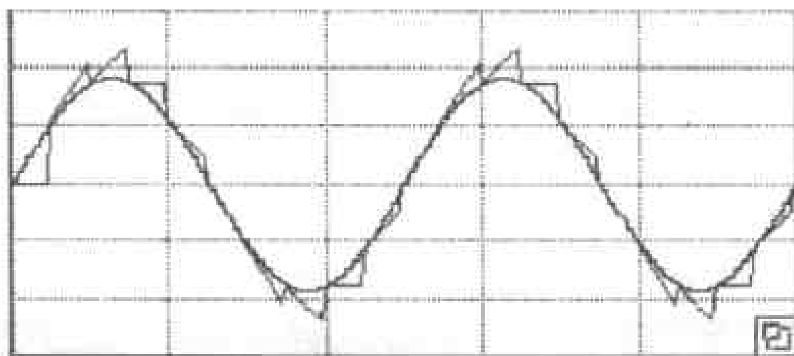


图 7.49　一阶保持对波形的影响



图 7.50　First-Order Hold 模块的参数对话框

3. 特性

- 数据类型：First-Order Hold 模块输入和输出双精度型信号；
- 采样时间：连续；
- 状态数：每一个输入元素有 1 个连续和 1 个离散；
- 可矢量化。

7.6.7　Unit Delay 模块

1. 功能描述

Unit Delay 模块将输入延迟并保持一个采样周期。若模块的输入是矢量，其所有元素都将被延迟一个采样周期。本模块相当于一个 z^{-1} 的时间离散算子。

编者提示：若需要一个非延迟采样-保持函数，请使用 Zero-Order Hold 模块。若需要一个大于一个采样周期的延迟，请使用 Discrete Transfer Fcn 模块。参看 Transport Delay 模块使用 Unit Delay 模块的例子。

2. 参数和对话框

Unit Delay 模块的参数对话框如图 7.51 所示。

- Initial condition：在模块未定义期间，模块的第一个仿真周期输出。请仔细选择该参数以减少不必要的输出。默认值为 0。
- Sample time：采样的时间周期。默认值为 1。

3. 特性

- 数据类型：Unit block 模块接受包括用户定义在内的数据型实或复信号。若输入信号的数据类型是由用户定义的，则初始条件必须为 0；
- 采样时间：离散；

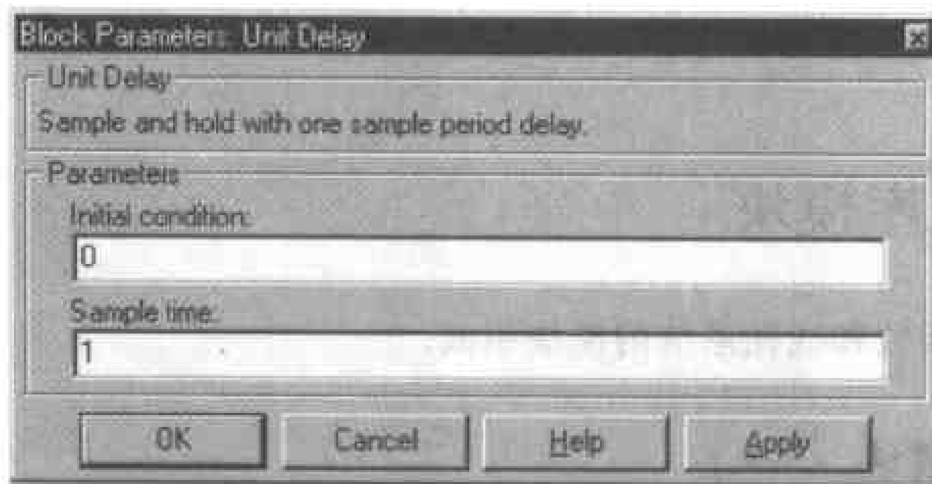


图 7.51 Unit Delay 模块的参数对话框

- 标量扩展：Initial condition 参数或输入；
- 状态数：从驱动或参数模块继承；
- 可矢量化。

7.6.8 Zero-Order Hold 模块

1. 功能描述

Zero-Order Hold 模块实现一个以指定采样率的采样与保持函数操作。模块接受一个输入，并产生一个输出，两者可以是标量或矢量。

本模块的工作原理是：离散化一个或多个信号，或以不同的采样率对信号进行重新采样。用户可以将模块用于需要建构采样但不要求其他更复杂的离散函数模块的场合。例如，可将本模块与 Quantizer 模块联合使用以建构一个单输入 A/D 变换器。

2. 参数和对话框

Zero-Order Hold 模块的参数对话框如图 7.52 所示。

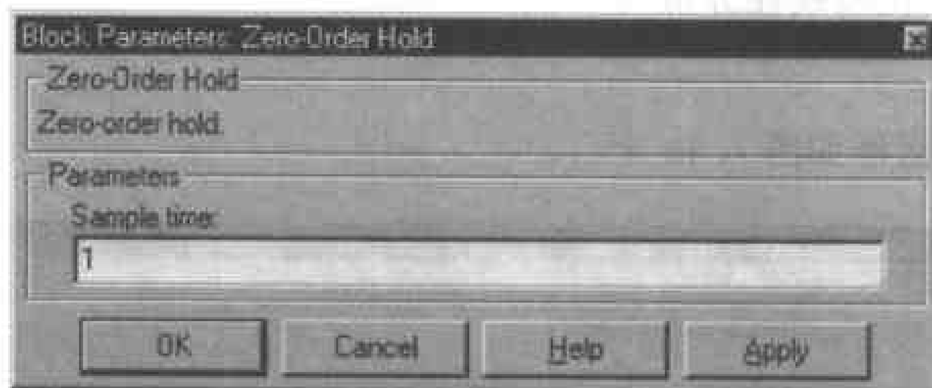


图 7.52 Zero-Order Hold 模块的参数对话框

- Sample time：采样的时间间隔。默认值为 1。

3. 特性

- 数据类型：Zero－Order Hold 模块接受任意数据类型的实或复信号；
- 直通输出；
- 采样时间：离散；

- 状态数：0；
- 可矢量化。

7.7 连续系统库模块

由描述标准线性函数和线性系统的模块组成。

7.7.1 Derivative 模块

1. 功能描述

Derivative 模块通过计算下面的公式来近似输入的导数：$\frac{\Delta u}{\Delta t}$

其中：Δu 为输入的变化量，Δt 为自前一次仿真时间步以来的时间变化量。模块有一个输入和一个输出。在仿真开始前的输入信号值假设为 0；模块的初始输出为 0。

结果的精度取决于仿真所采用的时间步长。步长越短，模块输出的曲线就越平滑，越精确。当输入快速变化时，该模块不像其他有连续状态的模块，它的 Solvers（仿真器）不会采用较小的步长。

如果输入为离散信号，当输入变化时，输入的连续导数是冲激；否则是 0。为得到离散型的离散导数，采用：

$$y(k)=\frac{1}{\Delta t}[u(k)-u(k-1)]$$

或 Z 变换：

$$\frac{Y(z)}{u(z)}=\frac{1-z^{-1}}{\Delta t}=\frac{z-1}{\Delta t\cdot z}$$

编者提示：用 linmod 对含有 Derivative 模块的模型进行线性化会遇到麻烦。欲知如何避免，请参看第 5 章“线性化及线性分析”。

2. 对话框

Derivative 模块对话框如图 7.53 所示。

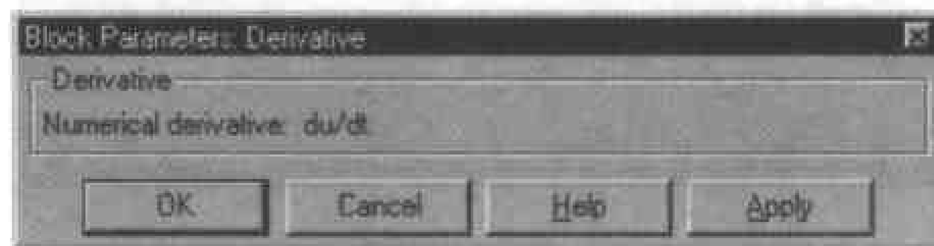


图 7.53　Derivative 模块对话框

3. 特性

- 数据类型：Derivative 模块输入和输出双精度型实信号；
- 直通输出；
- 采样时间：连续；
- 可矢量化；

- 过零检测。

7.7.2　Integrator 模块

1. 功能描述

本模块对其输入进行积分运算。用户通过本模块可以完成：

- 在对话框上定义初始条件或由输入完成；
- 输出模块的状态；
- 定义积分上、下限；
- 根据附加置位输入重置状态。

编者提示：若需建构一个纯数字系统时，请参看和使用"Discrete - Time Integrator 模块"。

(1) 定义初始条件。用户可在对话框中将初始条件定义为参数，或从外部信号输入：

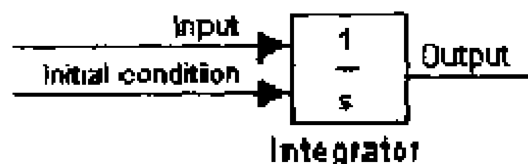


图 7.54　Integrator 模块的输入端口

- 将初始条件定义为参数。指定 Initial condition source 参数为 internal，并且在 Initial condition 参数栏中输入数值；
- 从外部源获得初始条件。将 Initial condition source 参数指定成 external，在模块输入(Input)端口的下方出现另一个输入端口，如图 7.54 所示。

(2) 使用状态端口。在下面两个情况下，用户必须使用状态端口而非输出端口：

- 当模块的输出通过置位端口或初始条件端口馈送到模块内时，会产生代数环。关于此种情况的例子，参看 bounce 模型。
- 当用户试图将模块的状态从一个条件执行子系统传递给另一个时，可能会产生定时问题。此种情况的例子，参看 clutch 模型。

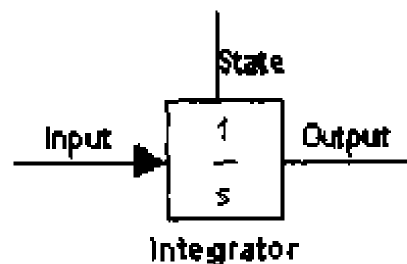


图 7.55　默认时 Integrator 模块的状态端口

用户可以通过状态端口而非输出端口来传递状态解决这类问题。尽管它们的值是相同的，但 Simulink 产生的时间会稍有不同，之所以如此，是为了避免用户模型出错。选择 Show state port 复选框输出模块状态。默认时，状态端口出现在模块的上方，如图 7.55 所示。

(3) 积分限。为避免输出超过已指定的电平，选中 Limit output 复选框，并在适当的参数栏内输入极限值。这样可使模块起定积分的作用。当输出超出极限值时，模块会停止积分以避免出现输出状态不定。在仿真期间内，用户可以更改极限值，但不能更改输出是否受限。模块输出由下列规则决定：

- 当积分小于下饱和限(Lower saturation limit)并且输入为负时，输出保持下饱和限值；
- 当积分处于下饱和限和上饱和限之间时，积分输出；
- 当积分大于上饱和限(Upper saturation limit)且输入为正时，输出保持上饱和限值。

选中 Show saturation port 复选框可以产生显示状态受限的时间。在模块输出端口的下方会出现一个饱和端口，如图 7.56 所示：

信号为下列三值之一：

- 1 表示上限起作用；
- 0 表示积分不受限；
- −1 表示下限起作用。

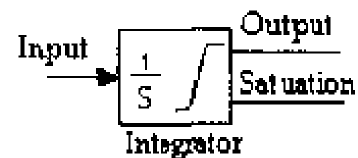


图 7.56 Integrator 模块的饱和端口

当复选框被选中后，模块就有三个过零检测点：一个用于检测模块进入上饱和限的时间；一个检测模块进入下饱和限的时间；再一个检测模块脱离饱和的时间。

(4) 设置模块的状态。模块可以根据基于外信号的指定初始条件重新设置其状态。为达到此目的，请选择 External reset 选项之一。一个触发端口就会出现在模块的输入端口下方，并显示触发类型，如图 7.57 所示：

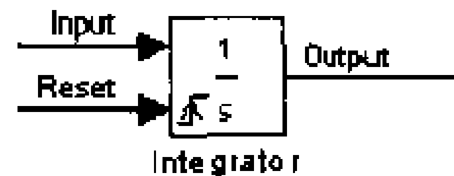


图 7.57 Integrator 模块的触发端口图标

- 选择 rising，在设置信号上升沿触发状态设置。
- 选择 falling，在设置信号下降沿触发状态设置。
- 选择 either，在上升沿或下降沿时触发。

设置端口是直通输出的。若输出以直接或通过直通输出模块反馈送入该端口，会造成代数循环。将模块状态送入状态端口即可解决。为访问模块的状态，请选择 Show state port 复选框。

(5) 指定模块状态的绝对容限。当用户的模型包含了多个数量差别很大的模块状态时，必须定义绝对容限，因为模型可能没有提供足够有效的误差控制。在 Absolute tolerance 参数中输入一个数值可以定义一个 Integrator 模块状态的绝对容限。若模块有不止一个状态，则输入的数值适用于所有状态。欲知误差控制的更多信息，请参看“误差容限”。

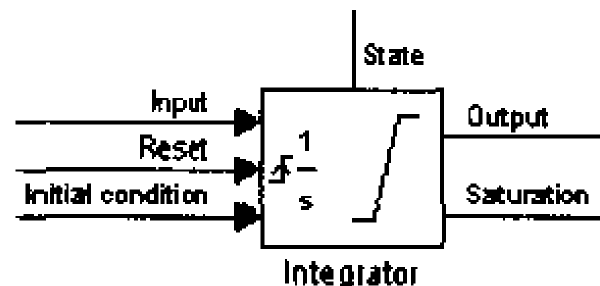


图 7.58 Integrator 模块端口图标全图

(6) 选中所有选项。当所有选项都被选中时，模块的图标看起来如图 7.58 所示。

2. 参数和对话框

Integrator 模块的参数对话框如图 7.59 所示。

- External reset：在设置信号的触发事件发生时(rising，falling 或 either)，根据初始条件设置模块状态。
- Initial condition source：若设为 internal 时，从 Initial condition 参数获得状态的初始条件；若设为 external 时，则从一个外部模块获得。
- Initial condition：状态的初始条件。设置 Initial condition source 参数。
- Limit output：若选中，输出状态将被限制在 Lower saturation limit 和 Upper saturation limit 参数之间的范围内。
- Upper saturation limit：积分上限。缺省值为 inf。
- Lower saturation limit：积分下限。缺省值为 −inf。
- Show saturation port：若选中，给模块加一个饱和输出端口。
- Show state port：若选中，给模块的模块状态加一个输出端口。
- Absolute tolerance：模块状态的绝对容限。

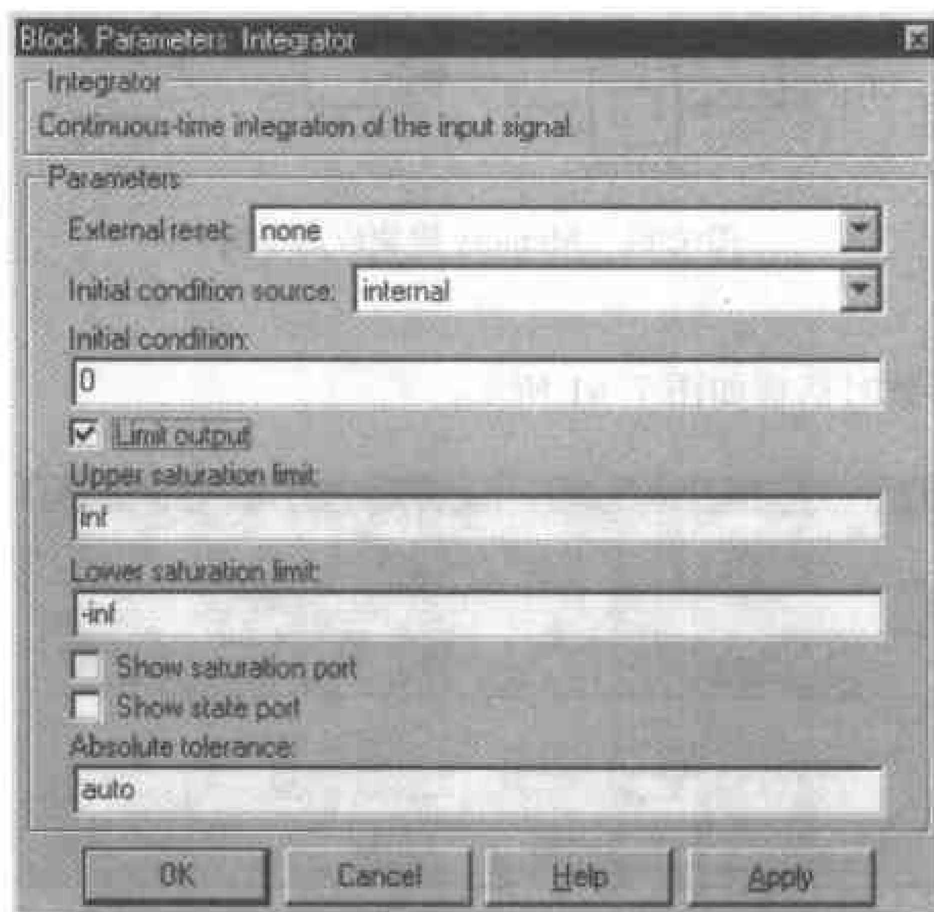


图 7.59 Integrator 模块的参数对话框

3. 特性

- 数据类型：Integrator 模块从数据端口接受和输出双精度型信号。外设置端口接受双精度型或布尔型信号；
- 直通输出：设置和外部初始条件源端口；
- 采样时间：从驱动模块继承；
- 标量扩展：所有参数；
- 状态数：从驱动模块继承或参数；
- 可矢量化；
- 过零检测：若选中 Limit output 选项，一个用于探测设置；一个用于探测上、下饱和限；一个探测脱离饱和时间。

7.7.3 Memory 模块

1. 功能描述

Memory 模块将前一个集成步的输入作为输出，相当于对前一个集成步内的输入进行采样-保持。

2. 应用举例

如图 7.60 所示的采样模型(应是较大模型的一部分)表明了如何显示一个仿真过程的步长。Sum 模块从当前时间中减去前一个由 Memory 模块产生的集成步时间。

编者提示：在使用 ode15s 和 ode113 时应避免使用 Memory 模块，除非模块的输入不发生变化。

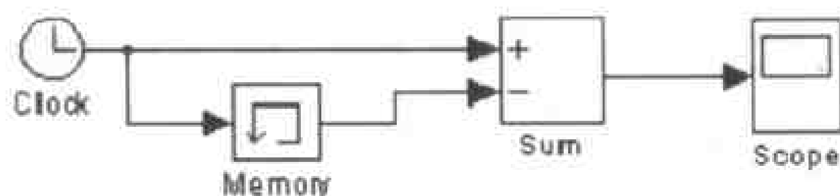


图 7.60　Memory 模块应用举例

3. 参数和对话框

Memory 模块的参数对话框如图 7.61 所示。

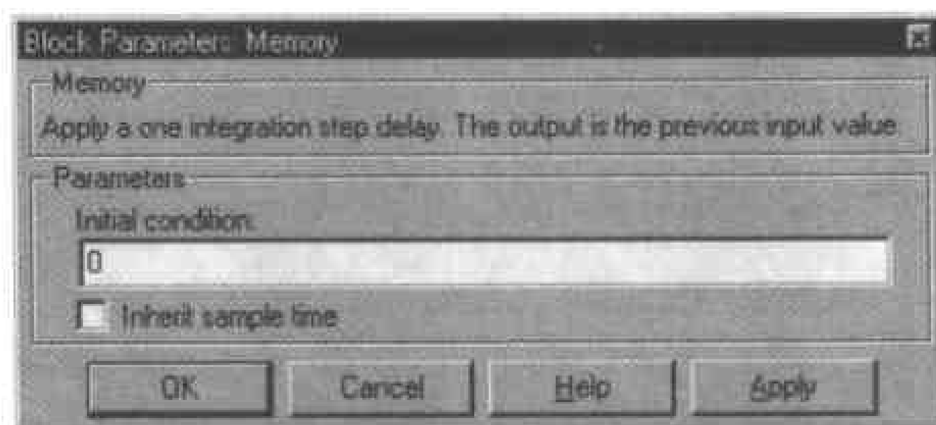


图 7.61　Memory 模块的参数对话框

- **Initial condition**：初始集成步的输出。
- **Inherit sample time**：选中此复选框使采样时间从驱动模块继承。

4. 特性

- 数据类型：Memory 模块接受包括用户定义的任意数值型(复或实)和数据型信号。若输入数据类型是用户定义的，则初始条件必须为 0；
- 采样时间：连续；若 Inherit sample time 复选框被选中，则继承；
- 标量扩展：Initial condition 参数；
- 可矢量化。

7.7.4 State-Space 模块

1. 功能描述

State-Space 模块实现一个由下式定义其特性的系统：

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

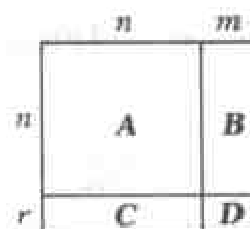


图 7.62　矩阵系数之特性

其中：x 为状态矢量，u 为输入矢量，y 为输出矢量。矩阵系数必须具有如图 7.62 所示之特性：

- A 必须是一个 $n \times n$ 矩阵，其中 n 为状态数；
- B 必须是一个 $n \times m$ 矩阵，其中 m 为输入数；
- C 必须是一个 $r \times n$ 矩阵，其中 r 为输出数量；
- D 必须是一个 $r \times m$ 矩阵。

模块接受一个输入，并产生一个输出。输入矢量的宽度由 B 和 D 矩阵的列数决定。输出矢量的宽度由 C 和 D 矩阵的行数决定。

为易于矩阵乘，Simulink 将含有 0 的矩阵转换为一个稀疏矩阵。

2. 参数和对话框

State-Space 模块的参数对话框如图 7.63 所示。

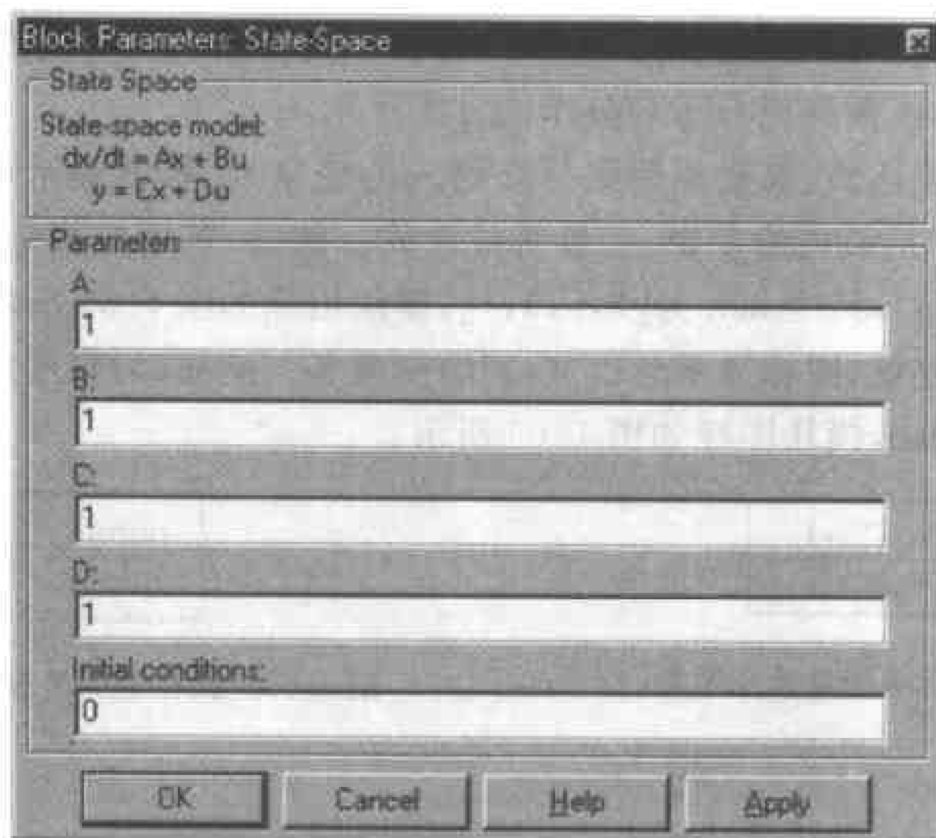


图 7.63　State-Space 模块的参数对话框

- A，B，C，D：矩阵系数。
- Initial conditions：初始状态矢量。

3. 特性

- 数据类型：State－Space 模块接受和输出双精度型实信号；
- 直通输出：仅当 $\boldsymbol{D}\neq 0$；
- 采样时间：连续；
- 标量扩展：初始条件；
- 状态数：取决于 $\boldsymbol{A}$ 的大小；
- 可矢量化。

7.7.5 Transfer Fcn 模块

1. 功能描述

Transfer Fcn 模块实现一个传递函数，其表达式为：

$$\boldsymbol{H}(s)=\frac{\boldsymbol{y}(s)}{\boldsymbol{u}(s)}=\frac{\boldsymbol{num}(s)}{\boldsymbol{den}(s)}=\frac{\boldsymbol{num}(1)s^{nn-1}+\boldsymbol{num}(2)s^{nn-2}+\cdots+\boldsymbol{num}(nn)}{\boldsymbol{den}(1)s^{nd-1}+\boldsymbol{den}(2)s^{nd-2}+\cdots+\boldsymbol{den}(nd)}$$

其中：$\boldsymbol{u}(s)$ 和 $\boldsymbol{y}(s)$ 分别表示输入和输出；nn 和 nd 分别为分子和分母系数的数量；**num** 和 **den** 包括了以 s 降幂次的分子和分母系数。**num** 可以是一个矢量或矩阵，但 **den** 必须是一个矢量。两者均由模块对话框中的参数来指定。分母的幂次必须大于或等于分子的幂次。

一个 Transfer Fcn 模块取一个标量输入。若模块传递函数的分子为一个矢量，模块输出

也是标量。然而，若分子是一个矩阵，传递函数把输入扩展为一个宽度等于分子行数的矢量。例如，一个双行分子使模块输入标量，输出矢量。输出矢量的宽度等于2。

初始条件预先设置为0。若用户需指定初始条件，使用tf2ss转换为状态方程的形式，然后使用State-Space模块。tf2ss的功用是给系统提供**A**、**B**、**C**和**D**矩阵。欲知详情，键入help tf2ss或参考"*Control System Toolbox User's Guide*"(《控制系统工具箱使用指南》)。

Transfer Fcn图标根据指定分子和分母的方式显示：

• 若指定为一个表达式、矢量或用圆括弧括入的变量，图标就显示带有指定系数和s的幂次。若用户指定一个括弧内的变量，变量就被计算。例如，若用户指定Numerator为[3,2,1]，Denominator为(den)，其中den为[7,5,3,1]，模块图标如图7.64所示。

• 若指定为一个变量，图标显示后跟"(s)"的变量名。例如，如果用户指定Numerator为num，Denominator为den，模块图标如图7.65所示。

$$\frac{3s^2+2s+1}{7s^3+5s^2+3s+1}$$

图7.64 图标随变量改变示意(1)

$$\frac{num(s)}{den(s)}$$

图7.65 图标随变量改变示意(2)

2. 参数和对话框

Transfer Fcn模块的参数对话框如图7.66所示。

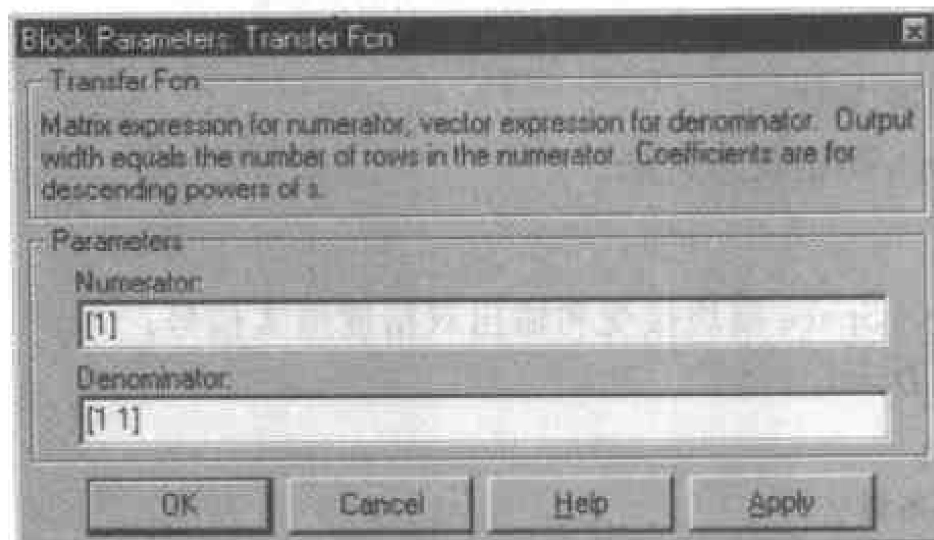


图7.66 Transfer Fcn模块的参数对话框

• Numerator：分子系数行矢量。可以指定多行矩阵产生多个输出。默认值为[1]。

• Denominator：分母系数行矢量。默认值[1 1]。

3. 特性

• 数据类型：Transfer Fcn模块接受和输出任意数据类型的信号；

• 直通输出：仅当Numerator和Denominator参数的长度相等时；

• 采样时间：连续；

• 状态数：Denominator长度−1；

• 矢量化：在模块将标量输入扩展为矢量输出情况下，传递函数的分子为一个矩阵。见模块的功能描述。

7.7.6 Transport Delay 模块 

1. 功能描述

Transport Delay 模块将输入延迟一个指定的时间量。可用于模拟一个时间延迟。在仿真启动后，当模块开始产生延迟输出时，模块输出 Initial input 参数值一直到仿真时间超过 Time delay 参数值为止。Time delay 参数必须是非负的。

在仿真期间，模块将输入点和仿真时间储存在一个缓冲器中，该缓冲器的容量由 Initial buffer size 参数指定。若输入点数超出缓冲器的容量，模块将配置额外的存储区，并且在表示需要总缓冲器容量的仿真之后，Simulink 显示信息。配置额外存储区会降低仿真速度，若仿真速度至关重要的话，请务必仔细定义该参数。对于长时间延迟，本模块会使用大量的存储区，特别对一个矢量化输入而言更是如此。

当要求在与存储输入值的时间不对应的时间输出时，模块在两个点之间进行线性插值计算。当延迟小于时间步长时，模块将从最后一个输出点进行外插计算，但这可能产生不精确的结果。因为本模块不具有直通输出，所以模块不能用当前输入计算输出值。作为举例，考虑一个固定步长的仿真，其步长为 1，当前时间在 $t = 4.5$。因为最近一个存储时间值在 $t = 4$，所以模块会自动进行前向外插计算。

Transport Delay 模块不能对离散信号进行插值计算。模块返回在 $t - t_{delay}$（当前时间-时间延迟）对应的离散值。

本模块不同于 Unit Delay 模块，Unit Delay 模块仅仅延迟和保持采样数输出。

使用 linmod 对一个包含 Transport Delay 模块的模型进行线性化处理是麻烦的。欲知解决方法，请参看第 5 章“线性化与线性分析”。

2. 参数和对话框

Transport Delay 模块的参数对话框如图 7.67 所示。

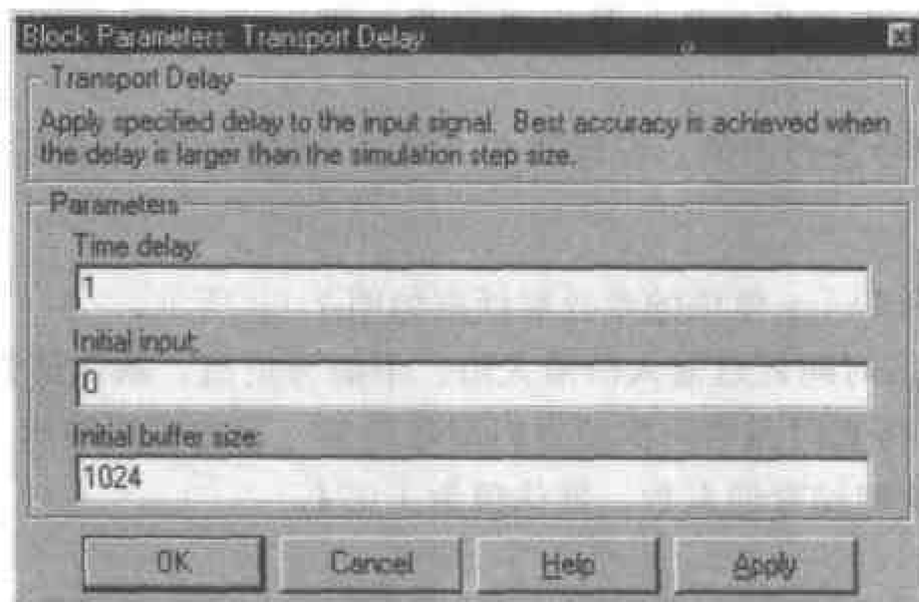


图 7.67 Transport Delay 模块的参数对话框

- Time delay：在输入传递到输出前，延迟输入信号的仿真时间量。
- Initial input：在开始仿真和 Time delay 之间模块产生的输出。
- Initial buffer size：储存点数的初始存储区配置。

3. 特性

- 数据类型：Transport Delay 模块接受和输出双精度型实信号；
- 采样时间：连续；
- 标量扩展：输入和除 Initial buffer size 外的参数；
- 可矢量化；
- 过零检测。

7.7.7 Variable Transport Delay 模块 

1. 功能描述

Variable Transport Delay 模块可用于模拟一个时间变化延迟。也可以用于构建一个管道传输系统，其马达抽取流体的速度是个变量。

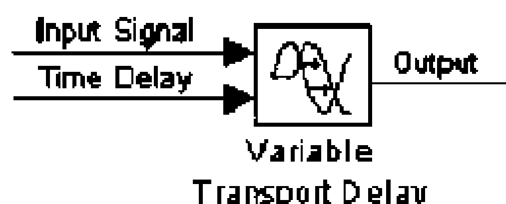


图 7.68 模块的输入端口

模块接受两个输入：第一个是通过模块的信号；第二个是时间延迟。如图 7.68 所示。

Maximum delay 参数定义输入的最大延迟时间。模块捕获超过该值的延迟值。Maximum delay 必须大于或等于零。若延迟时间为负值，模块将其强制为零，并给出警告。

在仿真期间，模块将时间/输入值对存入内部缓冲器中。在仿真开始后，模块输出参数值一直到仿真时间超过时间延迟输入值为止。然后，在每一个仿真步，模块输出当前仿真时间减去延迟时间所对应的信号。

当要求在不符合保存输入值的时间输出时，模块将在两点之间进行线性插值。若时间延迟小于仿真步长，模块对一个输出点进行外插。这可能造成精度降低。模块不能使用当前输入以计算其输出值，这是因为模块没有直通输出。为说明这一点，考虑一个固定步长为 1，当前时间为 $t=5$ 的仿真。若延迟为 0.5，模块需要产生在 $t=4.5$ 的点。由于最近储存的时间在 $t=4$，所以模块进行前向外插。

本模块不对离散信号进行插值。但返回在 $t-t_{delay}$（当前时间−延迟时间）的值。

2. 参数和对话框

Variable Transport Delay 模块的参数对话框如图 7.69 所示。

- Maximum delay：时间延迟输入的最大值。不能为负值。默认值为 10。
- Initial input：在仿真时间第一次超过时间延迟输入前的模块输出。默认值为 0。
- Buffer size：模块能储存的点数。默认值为 1 024。

3. 特性

- 数据类型：Variable Transport Delay 模块接受和输出双精度型实信号；
- 直通输出：时间延迟（第二个）输入；
- 采样时间：连续；
- 标量扩展：输入和除 Buffer size 以外的所有参数；
- 可矢量化。

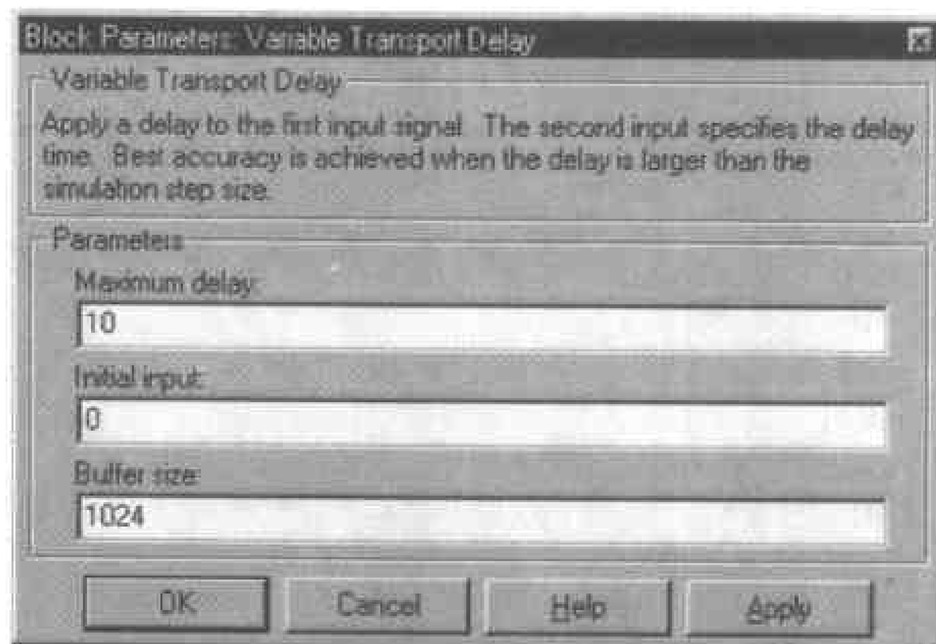


图 7.69 Variable Transport Delay 模块的参数对话框

7.7.8 Zero-Pole 模块 

1. 功能描述

本模块实现一个以 Laplace 算子 s 为变量的、零点、极点和增益可指定的系统。一个传递函数可以用零点-极点-增益形式表示。MATLAB 下单输入、单输出系统可表示为：

$$H(s)=K\frac{Z(s)}{P(s)}=K\frac{(s-Z(1))(s-Z(2))\cdots(s-Z(m))}{(s-P(1))(s-P(2))\cdots(s-P(n))}$$

其中，Z 代表为零点矢量；P 为极点矢量；K 为增益。Z 可以是一个矢量或矩阵；P 必须是一个矢量；K 可以是一个标量或矢量，其长度等于 Z 的行数。极点数必须大于或等于零点数。若极点和零点是复数，它们必须是复共轭对。

模块的输入和输出宽度等于零点矩阵的行数。

Zero-Pole 模块可根据参数的指定形式在其图标内显示传递函数：

• 若某一个参数被指定为一个表达式或一个矢量，图标就显示包含指定零点、极点和增益的传递函数。若用户指定一个在圆括弧内的变量，其值就被求出。

例如，假设将 Zeros 指定为[3,2,1]，Poles 为(poles)（poles 在工作空间被定义为[7,5,3,1]），Gain 为 gain，则显示的图标如图 7.70 所示。

gain(s-3)(s-2)(s-1) / (s-7)(s-5)(s-3)(s-1)

图 7.70 图标随参数变化示意(1)

gain*zeros(s) / poles(s)

图 7.71 图标随参数变化示意(2)

• 若指定为一个变量，图标显示后跟“(s)”的变量名。例如，假设用户将 Zeros 指定为 zeros，Poles 为 poles，Gain 为 gain，则显示的图标如图 7.71 所示。

2. 参数和对话框

Zero-Pole 模块的参数对话框如图 7.72 所示。

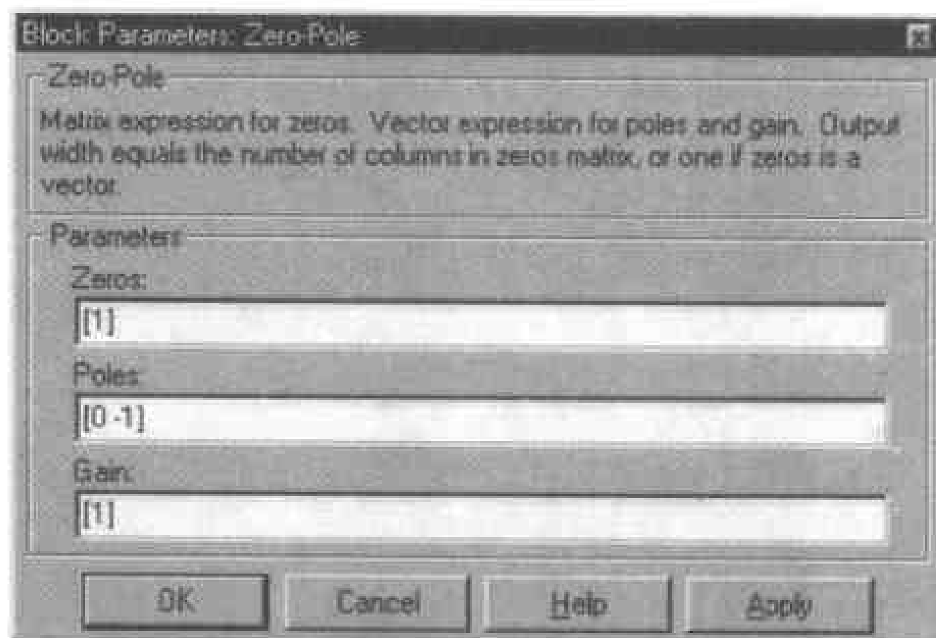


图 7.72 Zero-Pole 模块的参数对话框

- Zeros：零点矩阵。默认值为[1]。
- Poles：极点矢量。默认值为[0－1]。
- Gain：增益矢量。默认值为[1]。

3. 特性

- 数据类型：Zero-Pole 模块接受双精度型实信号；
- 直通输出：仅当 Poles 和 Zeros 参数的长度相等时；
- 采样时间：连续；
- 状态数：Poles 矢量的长度。

7.8 数学运算库模块

由描述数学运算的模块组成。基本函数模块请参考 Function & Table 库中的 Fun 模块。

7.8.1 Abs 模块 |u|

1. 功能描述

将输入信号的绝对值或模作为输出。

2. 对话框

Abs 模块的对话框如图 7.73 所示。

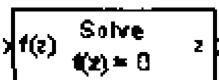


图 7.73 Abs 模块的对话框

3. 特性

- 数据类型:输入双精度实数或复数,输出双精度实数;
- 直通输出;
- 采样时间:从驱动模块继承;
- 矢量化;
- 过零检测。

7.8.2 Algebraic Constraint 模块



1. 功能描述

该模块将输入 f(z) 强制置为零并输出 z。模块输出导致输入为零的值。输出可以通过某反馈路径影响输入。用户能够对索引 1 的 differential/algebraic systems (DAEs)指定其条件约束方程。缺省时,"Initial guess"参数为零。用户可以通过给该参数设置更接近于求解值的 z ,来提高环路仿真器求解的效率。

2. 应用举例

求解下列方程组的模型:

$$z2 + z1 = 1$$

$$z2 - z1 = 1$$

如图 7.74 显示模块(Display)所示,解是 $z2 = 1$, $z1 = 0$。

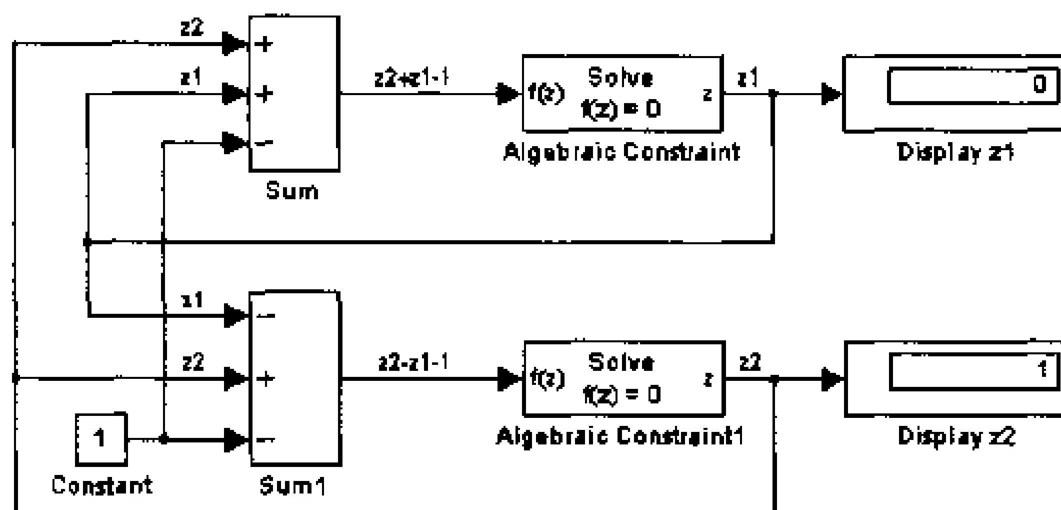


图 7.74 模块应用举例

3. 参数和对话框

Algebraic Constraint 模块的参数对话框如图 7.75 所示。

- **Initial guess**:求解初始预测值。默认值为 0。

4. 特性

- 数据类型:输入和输出结尾双精度实数;
- 直通输出;
- 采样时间:取决于输入模块;

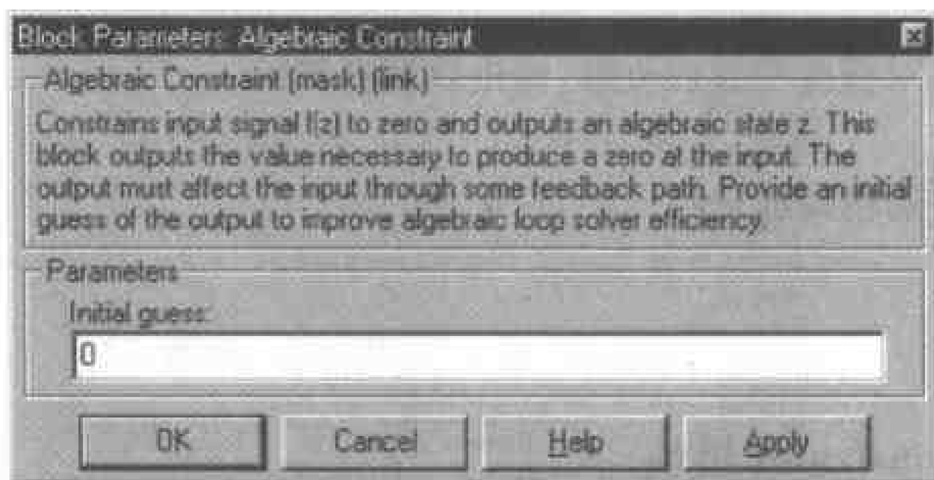


图 7.75　Algebraic Constraint 模块的参数对话框

• 可矢量化。

7.8.3　Combinatorial Logic 模块

1. **功能描述**

该模块用于建立可编程逻辑阵列(PLAs)、逻辑电路、判断电路和其他布尔公式的标准真值表。用户可以将该模块与 Memory 模块合起来实现有限态机或触发器。可将定义所有可能的模块输出的矩阵指定为 Truth table 的参数。矩阵每一行包含对于不同输入元素组合相应的输出。但必须指定将每一种输入组合的输出。列数是模块输出的数量。

输入数量与行数之间的关系是:行数=2^(输入的个数)。

Simulink 从计算输入矢量元素的行指数获得行。Simulink 通过建立一个二进制数列来计算指数,在二进制数中,零输入矢量元素为零,而非零输入矢量元素均为 1,结果最后再加 1。对于有 m 个元素的输入矢量 $\boldsymbol{u}$,

$$行指数=1+\boldsymbol{u}(m)*2^0+\boldsymbol{u}(m-1)*2^1+\ldots+\boldsymbol{u}(1)*2^{m-1}$$

2. **应用举例**

例一:双输入'与'函数。

此例建立一双输入'与'函数:当两个输入元素均为 1 时,输出为 1,否则为 0。为实现这种函数,指定 Truth table 参数值为[0; 0; 0; 1]。规定 Combinatorial Logic 模块的输入和输出的模型可能如图 7.76 所示。



图 7.76　Combinatorial Logic 模块应用举例

表 7.10 给出了产生输出的输入组合。对应于表中的列输入 1,输入信号标以"输入 1"。类似地,输入信号"输入 2"对应于相同名称的列。这些值的组合决定输出列中哪一个结果作为模块的输出。

例如,假设输入矢量为[1 0],涉及第三行(2^1*1+1)。则输出为 0。

表 7.10 输入/输出组合

行	输入 1	输入 2	输出
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

例二：电路。

该采样电路有三个输入：二个输入（a 和 b）和一个选通输入（c）；有两个输出，一个选通输出（c′）和一个求和输出（s）。表 7.11 为该电路的真值表和与不同输入组合相关的输出。

表 7.11 输入/输出组合

Inputs			Outputs	
a	b	c	c′	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

为借助 Combinatorial Logic 模块实现该加法器，须输入由列 c′和 s 构成的 8×2 矩阵作为 Truth table 的参数。时序电路（与状态有关）也可以借助 Combinatorial Logic 模块来实现，但必须增加模块状态的输入并将模块的输出反馈到此状态输入中。

3. 参数和对话框

Combinatorial Logic 模块的参数对话框如图 7.77 所示。

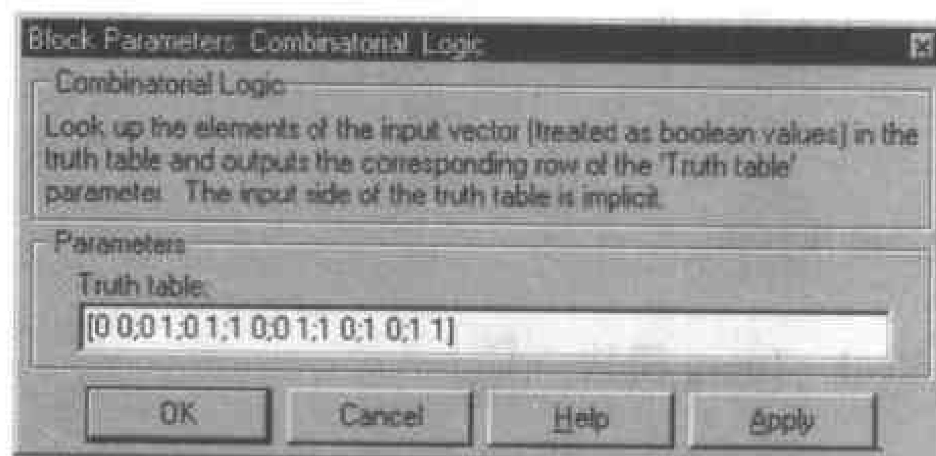


图 7.77 Combinatorial Logic 模块的参数对话框

- **Truth table**：输出矩阵。每列对应于输出矢量的一个元素，而每行对应于真值表的一行。

4. 特性

• 数据类型:Combinatorial Logic 模块接受布尔型或双精度的实信号,并输出同类型的信号。真值表的元素可以是布尔型或双精度型。如果输入元素是双精度型的,它们可以是任何值,并不是非布尔值(0 或 1)不可。如果真值表的数据类型与输出信号的不同,Simulink 在计算之前会将真值表转换为输出的数据类型。
• 直通输出;
• 采样时间:从驱动模块继承;
• 矢量化:输出宽度为 Truth table 参数的列数。

7.8.4 Complex to Magnitude－Angle 模块 

1. 功能描述

Complex to Magnitude-Angle 模块接受双精度复信号。它输出的输入信号幅值和/或相角取决于 Output 参数的设置。输出为双精度实信号。输入可以是矢量复信号,在此种情况下,输出也是矢量。幅值输出信号矢量包含了对应复信号输入之各个元素的幅值。类似地,相角输出包含了输入元素的相角。

2. 参数和对话框

Complex to Magnitude-Angle 模块的参数对话框如图 7.78 所示。

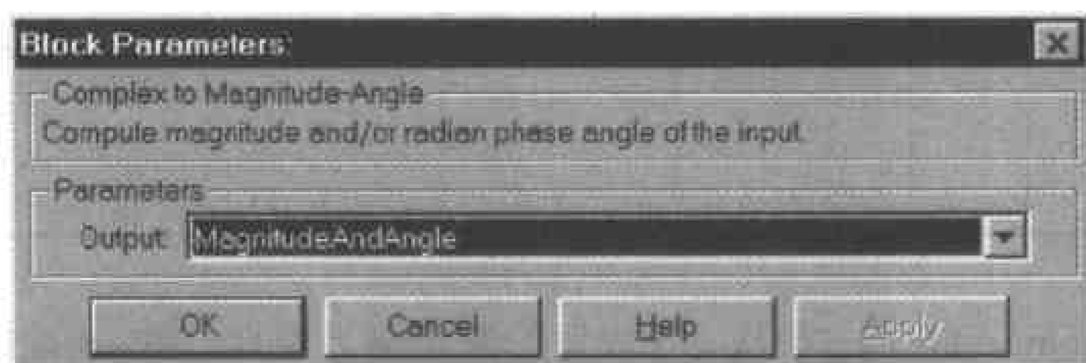


图 7.78　Complex to Magnitude-Angle 模块的参数对话框

Output:确定模块的输出。选择如下之一:
• MagnitudeAndAngle (输出输入信号的幅值和弧度角);
• Magnitude (输出输入信号的幅值);
• Angle(输出输入信号的弧度角)。

3. 特性

• 直通输出;
• 采样时间:从驱动模块继承。

7.8.5 Complex to Real－Imag 模块 

1. 功能描述

Complex to Real－Imag 模块接受双精度的复信号。输出的输入信号实部和(或)虚部取决于 Output 参数的设置。输出为双精度型实数。输入可以是矢量复信号,在此种情况下,输

出也是矢量。实信号矢量包含了对应复信号输入之各个元素的实部。类似地，虚部输出包含了输入元素的虚部。

2. 参数和对话框

Complex to Real-imag 模块的参数对话框如图 7.79 所示。

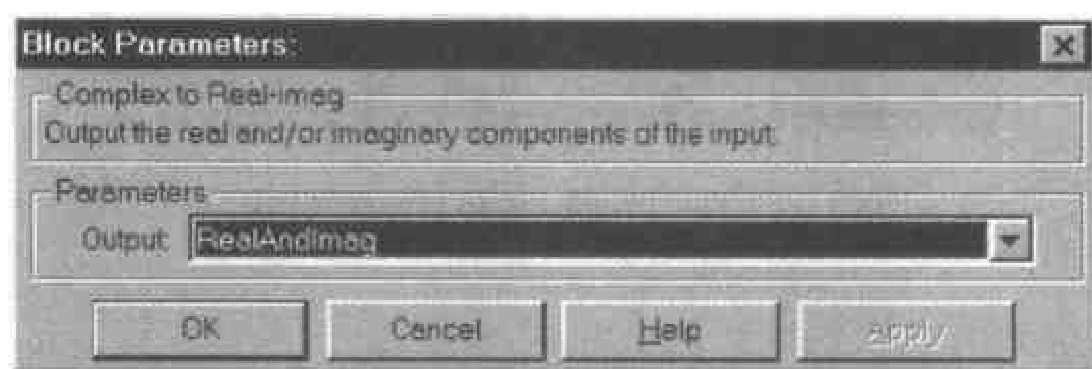


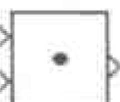
图 7.79　Complex to Real-imag 模块的参数对话框

Output：确定模块的输出。选择如下之一：

- RealAndImag(输出输入信号的实部和虚部)；
- Real(输出输入信号的实部)；
- Imag (输出输入信号的虚部)。

3. 特性

- 直通输出；
- 采样时间：从驱动模块继承。

7.8.6 Dot Product 模块 

1. 功能描述

Dot Product 模块对两个输入矢量进行点积运算。标量输出，y 等于 MATLAB 运算：

$$y = \boldsymbol{u}1' * \boldsymbol{u}2$$

其中：$\boldsymbol{u}1$ 和 $\boldsymbol{u}2$ 表示矢量输入。若两个输入均为矢量，长度必须相等。输入矢量的元素可以是双精度实数或复数信号。输出信号的数据类型(实或复)取决于输入。

表 7.12　输入组合与输出对照

输入 1	输入 2	输出
实数	实数	实数
实数	复数	复数
复数	实数	复数
复数	复数	复数

编者提示：若进行不求和的积运算，请用 Product 模块。

2. 对话框

Dot Product 模块的参数对话框如图 7.80 所示。

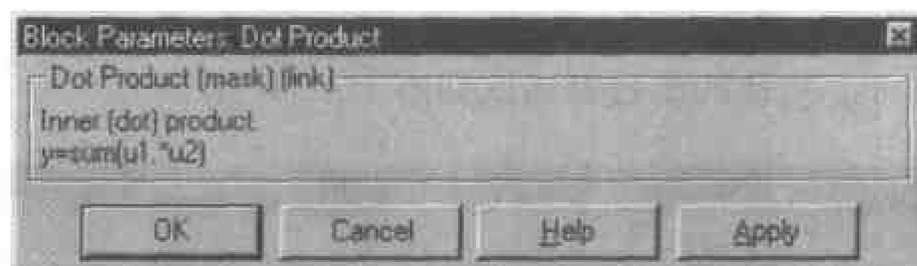
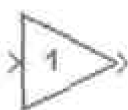


图 7.80 Dot Product 模块的参数对话框

3. 特性

- 数据类型:Dot Product 模块接受和输出双精度型信号;
- 直通输出;
- 采样时间:从驱动模块继承;
- 标量扩展;
- 状态数:0
- 可矢量化。

7.8.7 Gain 模块 

1. 功能描述

Gain 模块将模块的输入乘上一个指定的常数、变量或表达式后输出。用户可输入数值或变量增益或表达式。

如果模块足够大,Gain 的图标显示 Gain 参数域中输入的值。如果增益是变量,则显示变量名。若 Gain 参数过长,则显示-K-。

编者提示:若需用矩阵相乘,请使用 Matrix Gain 模块。4.X 版本已将此模块归入 Gain 模块。另外,使用滑块控制可以在仿真中修改增益,参看“Slider Gain 模块”。

2. 参数和对话框

Gain 模块的参数对话框如图 7.81 所示。

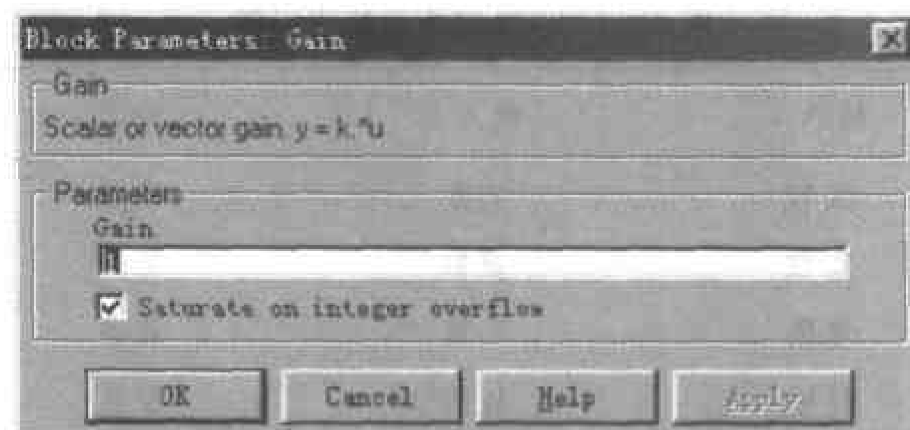


图 7.81 Gain 模块的参数对话框

• Gain：增益，可定义为标量，矢量，变量名或表达式。默认值为 1。若没有定义，则 Gain 参数为双精度型。

• Saturate on integer overflow：若被选中，模块输出在整数溢出时饱和。特别是，当输出的数据类型是整数型，模块的输出是输出类型或计算结果可表示的最大值，但绝对值小于该值。若该选项未选中，Simulink 将采取 Simulation Parameters 对话框 Diagnostics 页中 Data overflow 事件选项指定的动作。请参看第 4 章“Diagnostics 选项”。

3. 特性

• 数据类型：Gain 模块接受除布尔型以外的任意数据类型的实或复值标量或矢量，输出与输入数据类型相同。输入矢量的各个元素的数据类型必须相同。模块的 Gain 参数的数据类型同上。Gain 模块遵守下列规则：

① 如果输入是实数且增益是复数，则输出是复数；

② 如果 gain 参数的数据类型不同于输入信号且可被输入的数据类型所替换，则 Simulink 在计算其输出前，将增益转换为输入信号的数据类型。否则，Simulink 暂停仿真并给出错误信息。例如，若输入信号的数据类型是 uint8 而增益是－1，就会导致错误。若将增益参数改为输入信号的数据类型而导致精度损失的话，Simulink 会给出警告但继续仿真；

③如果输出是整数型且模块的 Saturate on integer overflow 选项有效，则当输出超出模块所规定的最大输出值时，输出饱和。例如，若 Gain 输出数据类型为 int8，当计算结果大于 128 时，输出就为 128，若计算结果小于－128，则输出－128。

• 直通输出；
• 采样时间：从驱动模块继承；
• 标量扩展：输入和 Gain 参数；
• 可矢量化。

7.8.8 Logical Operator 模块 AND

1. 功能描述

模块对输入进行一种指定的逻辑运算：“与”(AND)，“或”(OR)，“与非”(NAND)，“或非”(NOR)，“异或”(XOR)以及“非”逻辑。输出取决于输入的数值、矢量长度，以及所选的运算符。“真”时输出为 1，“假”时输出为 0。模块图标显示所选运算符。

• 对于超过一个的输入，模块对所有输入进行运算。若输入为矢量，运算将在相应的元素间进行，产生一个矢量输出。

• 对单个矢量输入，模块对矢量的所有元素进行运算(“非”逻辑除外)。“非”逻辑仅接受一个输入，输入可以是标量或单元素矢量。若输入是一个矢量，输出为相同长度并包含了输入矢量所有元素的逻辑补码。

当模块被设定为一个多输入“异或”门时，模块执行符合 IEEE 之逻辑原理标准的模二加运算。

2. 参数和对话框

Logical Operator 模块的参数对话框如图 7.82 所示。

• Operator：将要用于模块输入的逻辑算子。默认时的选择算子见上面所列。

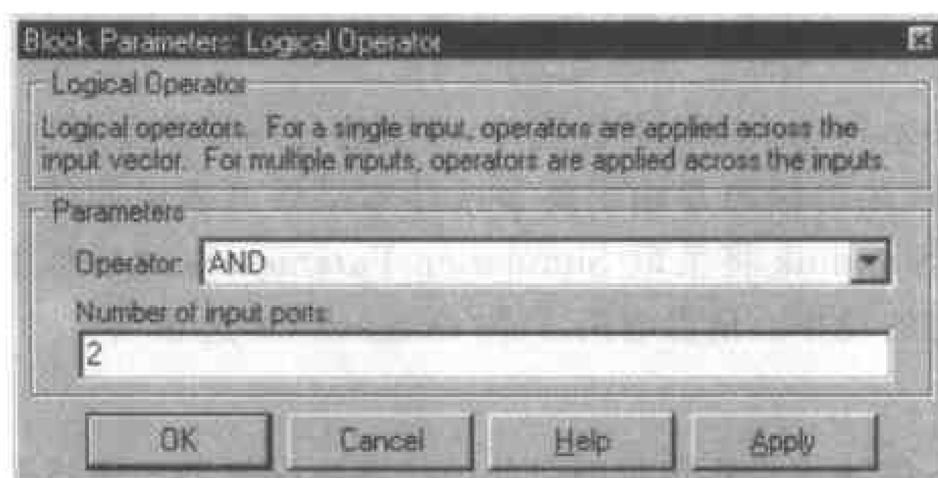


图 7.82　Logical Operator 模块的参数对话框

- Number of input ports：模块输入的数量。应适合所选算子的要求。

3. 特性

- 数据类型：Logical Operator 模块从输入端口接受布尔型信号，除非激活了布尔型兼容模式(参看第 3 章中有关“激活严格布尔型检查”的内容)。在此情况下，模块也接受双精度型输入。非零值双精度型输入均被当作“真”(1)对待，零值输入当作“假”(0)。所有输入的数据类型必须相同。模块输出的数据类型与输入相同。
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：输入；
- 可矢量化。

7.8.9　Magnitude-Angle to Complex 模块

1. 功能描述

Magnitude-Angle to Complex 模块能将一个幅度和(或)一个相角信号变换为复信号输出。输入必须是双精度型实信号。相角单位假设为弧度。复信号输出是双精度型。

输入可以是两个相同大小的矢量，或者，一个是矢量，另一个是标量。如果模块的输入是矢量，则输出是复信号矢量。一个幅度输入矢量的元素映射到对应复输出元素的幅度，类似地，一个相角输入矢量映射到复输出信号的相角。若一个输入是一个标量，则它映射到所有复输出信号的对应成分(幅度或相角)上。

2. 参数和对话框

Magnitude-Angle to Complex 模块的参数对话框如图 7.83 所示。

- Input：指定输入的种类：幅度输入、相角输入或两者。
- Angle (Magnitude)：若输入是一个单值信号，指定输出信号的常量幅度。若输入是一个幅度，则此参数指定以弧度为单位的输出信号的常量相角。

编者提示：请参看“Complex to Magnitude-Angle 模块”。

3. 特性

- 直通输出；

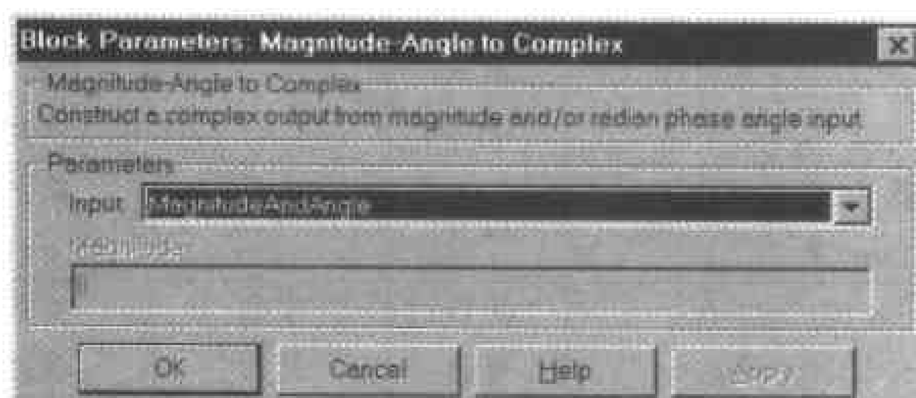


图 7.83 Magnitude-Angle to Complex 模块的参数对话框

- 采样时间：从驱动模块继承；
- 标量扩展：输入（两个输入）；
- 可矢量化。

7.8.10 Math Function 模块

1. 功能描述

Math Function 模块可进行多种常用数学函数运算。

用户可以从 Function 表中选择如下函数之一：exp，log，10u，log10，square，sqrt，pow，reciprocal，hypot，rem，mod 等。模块输出为用这些函数对输入进行计算以后的结果。函数名在模块图标上显示。Simulink 自动画出适当数量的输入端口。

编者提示：当用户要矢量化时，应使用本模块而不是 Fcn 模块。

Math Function 模块接受双精度型复或实信号，或信号矢量。输出信号的数据类型取决于 Output signal type 参数的设置。

2. 参数和对话框

Math Function 模块的参数对话框如图 7.84 所示。

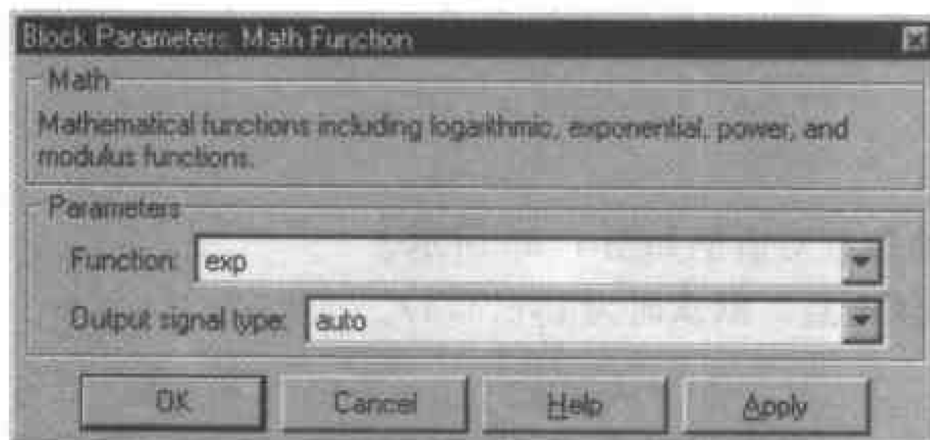


图 7.84 Math Function 模块的参数对话框

- Function：指定数学函数。
- Output signal type：对话框允许用户将 Math Function 模块的输出数据类型选择为实数、复数或自动。

表 7.13 输入/输出的数据类型组合

	输入(Input)	输出信号的数据类型(Output Signal Type)		
函数(Function)	信号	自动 auto	实数 real	复数 complex
Exp, log, log², log10, square, sqrt, pow, reciprocal, conjugate	实数	实数	实数	复数
	复数	复数	误差	复数
magnitude squared	实数	实数	实数	复数
	复数	实数	实数	复数
hypot, rem, mod	实数	实数	实数	复数
	复数	误差	误差	误差

3. 特性

- 直通输出;
- 采样时间:从驱动模块继承;
- 标量扩展:输入,仅当函数要求两个输入时;
- 可矢量化。

7.8.11 Matrix Gain 模块 K

1. 功能描述

本模块提供一个矩阵增益。它用指定的矩阵与输入矢量相乘,然后输出:

$$y = \boldsymbol{K}u$$

其中,$\boldsymbol{K}$ 为增益,u 为输入。

若指定的矩阵为 m 行和 n 列,则模块的输入应是一个长度为 n 的矢量,输出是长度为 m 的矢量。

模块图标始终显示 $\boldsymbol{K}$。若指定的矩阵包含了 0,Simulink 就把矩阵增益转换为提高乘法效率的稀疏矩阵。

2. 参数和对话框

Matrix Gain 模块的参数对话框如图 7.85 所示。

- Gain matrix:矩阵增益。默认时为 eye(3,3)。

3. 特性

- 数据类型:Matrix Gain 模块接受和输出双精度型实信号;
- 直通输出;
- 采样时间:连续
- 状态数:0;
- 可矢量化。

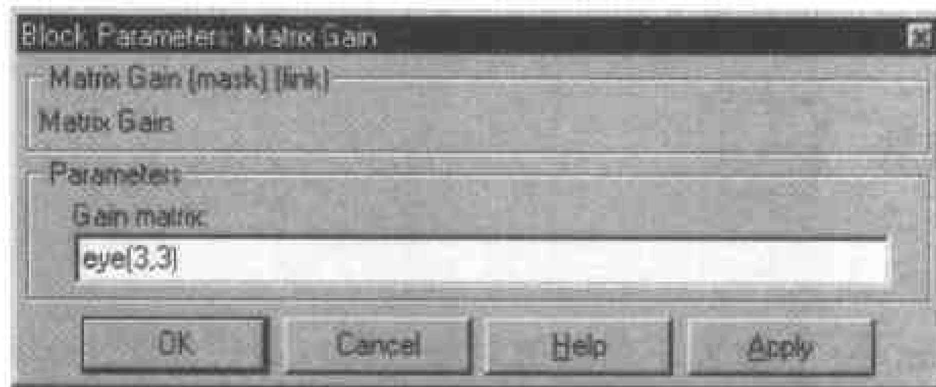


图 7.85 Matrix Gain 模块的参数对话框

7.8.12 MinMax 模块

1. 功能描述

MinMax 模块将输入的最小或最大值的元素或所有元素作为输出。用户可以通过选择 Function 参数表中的函数来确定欲使用的函数。

若模块有一个输入端口，模块将输入矢量的最小值元素或最大值元素用一个标量输出。

若模块的输入端口多于一个，模块将对输入矢量的各元素逐一进行比较，模块输出矢量的各元素即为输入矢量各元素的比较结果。

2. 参数和对话框

MinMax 模块的参数对话框如图 7.86 所示。

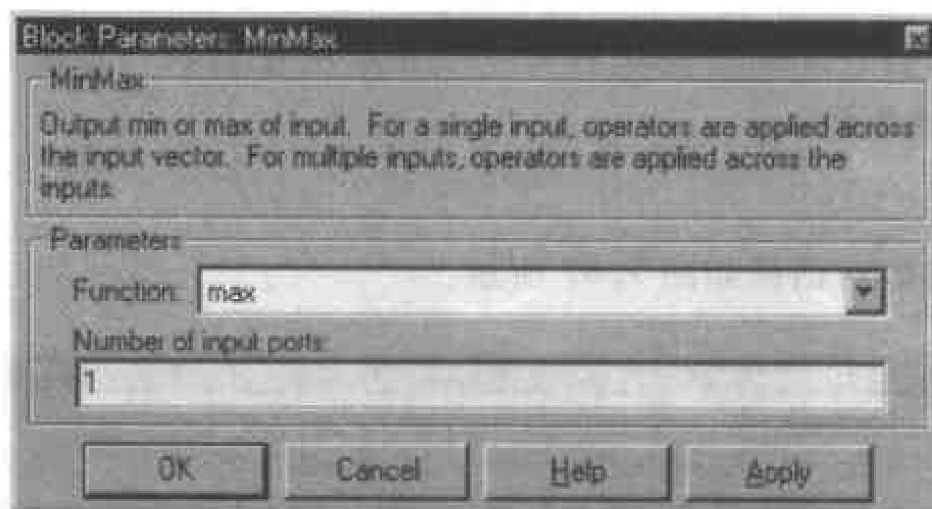


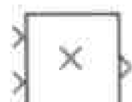
图 7.86 MinMax 模块的参数对话框

- Function：选择输入的函数(min 或 max)。
- Number of input ports：模块输入数。

3. 特性

- 数据类型：MinMax 模块接受和输出双精度型实信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：所有输入；
- 可矢量化；

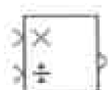
- 过零检测:探测最小值和最大值。

7.8.13　Product 模块 

1. 功能描述

根据 Number of inputs 参量的下列取值对输入进行乘法或除法运算:

- 在输入端口旁显示适当的符号。例如,下面的模块图标就是在输入的参数值为 * /时的结果。



- 若值是大于1的标量,模块把所有输入相乘。若输入是个矢量,模块输出为矢量各元素之乘积。若所有输入皆为标量,则输出也为标量。对任何一个有 n 个输入的模块来说,若输入是矢量,则产生的输出之各元素:

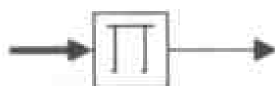
$$y_i = u1_i \times u2_i \times \cdots \times un_i$$

- 若值为1,模块输出为输入矢量各元素之标量积。

$$y = \prod u_i$$

2. 应用举例

如下模型表明了 Product 模块的使用方法。粗实线输入信号表示输入是一个矢量。



若必要,Simulink 会自动调整模块的尺寸以显示所有输入。若输入数发生了变化,端口将从模块的底部添加或删除。

3. 参数和对话框

Product 模块的参数对话框如图 7.87 所示。

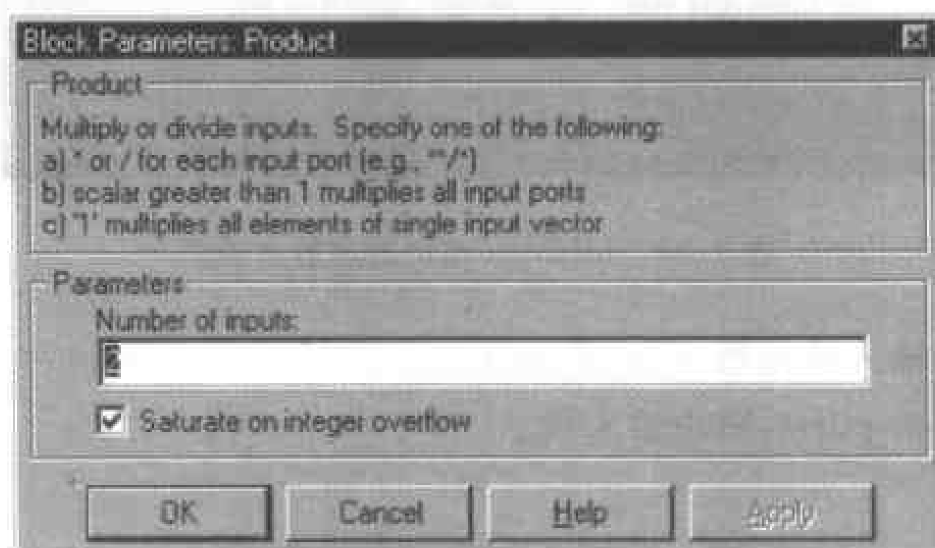


图 7.87　Product 模块的参数对话框

- **Number of inputs**:模块的输入数或"*"和"/"符号的组合。默认值为2。
- **Saturate on integer overflow**:若选中此选项,Product 模块的输出在整数溢出时饱和。

实际上，若输出数据类型是整数的话，模块输出是模块输出数据类型可表示的最大数或计算的输出，而不论两者之一的绝对值谁更小些。若该选项未选中，Simulink 将采取 Simulation Parameters 对话框中 Diagnostics 栏之 Data overflow 事件选项指定的行动（参看第5章"Diagnostics 选项"）。

4. 特性

- 数据类型：Product 模块接受任意数据类型的实或复信号。所有输入信号的数据类型必须相同，所以输出信号的数据类型也与输入相同；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展；
- 可矢量化。

7.8.14 Real-Imag to Complex 模块



1. 功能描述

Real－Imag to Complex 模块将实部和(或)虚部输入转换成一个复信号输出。输入必须是双精度型实信号。

输入可以是两个相同长度的矢量或者一个是矢量，另一个是标量。若模块的输入是矢量，输出就是一个复信号矢量。一个实部信号输入的元素映射到对应输出复信号元素的实部。类似地，一个虚部输入矢量则映射到输出复信号的虚部。若输入是一个标量，就映射到所有输出复信号的相应组分(实或虚)。

2. 参数和对话框

Real-Imag to Complex 模块的参数对话框如图 7.88 所示。

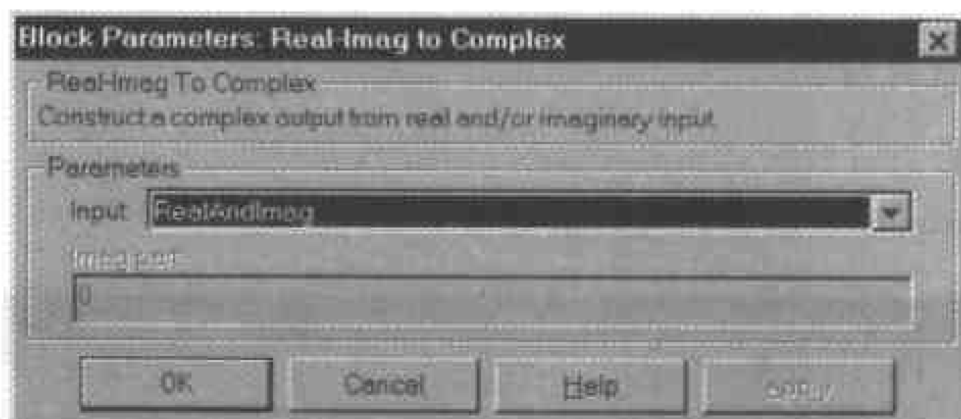


图 7.88 Real-Imag to Complex 模块的参数对话框

- **Input**：指定输入的种类：实部输入、虚部输入或两者。
- **Real (Imag) part**：若输入是一个实部信号，此参数指定输出信号的虚部值，若输入是一个虚部信号，参数指定输出信号的实部值。注意此栏标题的变化。

3. 特性

- 直通输出；

- 采样时间：从驱动模块继承；
- 标量扩展：参数（仅当函数要求双输入时）；
- 可矢量化。

7.8.15 Relational Operator 模块

1. 功能描述

Relational Operator 模块对它的两个输入进行相关性运算并根据表 7.14 产生输出。

表 7.14　算符与输出对应表

算符	输　出
==	TRUE，若第一个输入与第二个输入相同
~=	TRUE，若第一个输入不等于第二个输入
<	TRUE，若第一个输入小于第二个输入
<=	TRUE，若第一个输入小于或等于第二个输入
>=	TRUE，若第一个输入大于或等于第二个输入
>	TRUE，若第一个输入大于第二个输入

若运算结果为"TRUE"，则输出为 1；若结果为"FALSE"则输出为 0。用户可以将输入指定为标量、矢量或一个标量和一个矢量的组合，其输出情况分别为：

- 若输入是标量，输出是一个标量；
- 对矢量输入，输出是一个矢量，其中的元素为输入矢量各元素比较之结果；
- 对混合标量矢量输入，输出为一个矢量，其元素分别为标量与对应矢量元素比较之结果。

模块图标显示所选算符。

2. 参数和对话框

Relational Operator 模块的参数对话框如图 7.89 所示。

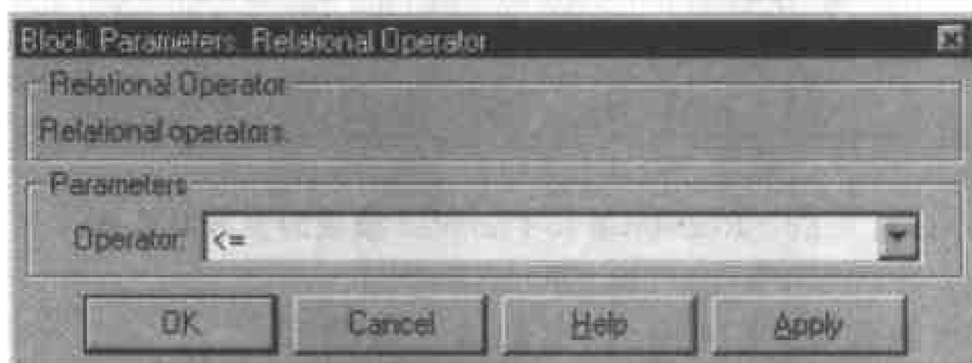


图 7.89　Relational Operator 模块的参数对话框

- **Operator**：用于模块输入的相关性算符。

3. 特性

- 数据类型：模块接受任意数据类型的实信号，输出一般为逻辑型信号。只有当非逻辑兼容性被激活（参看第 3 章“激活严格逻辑型校验”）时，模块才会输出双精度型信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：输入；
- 可矢量化；
- 过零检测：探测输出变化的时间。

7.8.16 Rounding Function 模块 floor

1. 功能描述

Rounding Function 模块实现常用的数学取整函数。

用户可以从 Function 表中选择一种函数：向下取整（floor）、向上取整（ceil）、四舍五入（round）和就近取整（fix）。模块的输出为函数对一个输入或多个输入的运算结果。选取的函数名会显示在模块图标中。

当用户需要矢量输出时，请使用 Rounding Function 模块代替 Fcn 模块，这是因为 Fcn 模块只能产生标量输出。

2. 参数和对话框

Rounding 模块的参数对话框如图 7.90 所示。

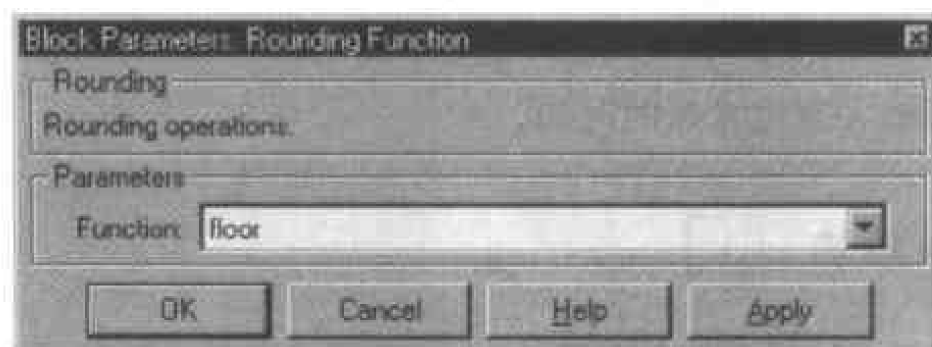


图 7.90 Rounding 模块的参数对话框

- **Function**：取整函数。

3. 特性

- 数据类型：Rounding Function 模块接受和输出双精度型实或复信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.8.17 Sign 模块

1. **功能描述**

Sign 模块指出输入的符号：

- 当输入大于 0 时，输出为 1；
- 当输入等于 0 时，输出为 0；
- 当输入小于 0 时，输出为 −1。

2. **对话框**

Sign 模块的对话框如图 7.91 所示。

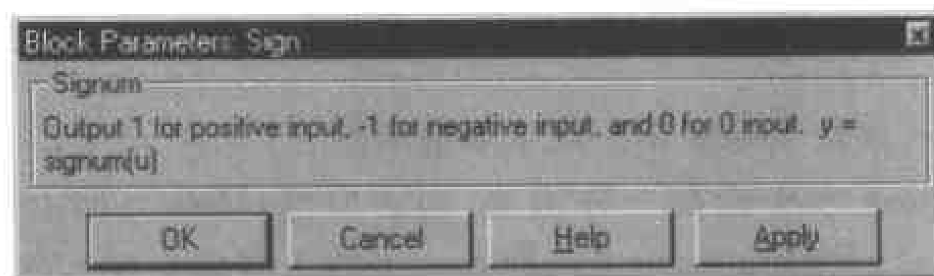


图 7.91 Sign 模块的对话框

3. **特性**

- 数据类型：Sign 模块接受和输出双精度型信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 可矢量化；
- 过零检测：探测输入通过零点的时间。

7.8.18 Slider Gain 模块

1. **功能描述**

Slider Gain 模块允许用户在仿真期间使用滑尺改变标量增益值。模块接受一个输入并产生一个输出。

2. **对话框**

Slider Gain 模块的参数对话框如图 7.92 所示。



图 7.92 Slider Gain 模块的参数对话框

- Low：增益下限。默认值为 0。
- High：增益上限。默认值为 2。

编辑栏从左到右分别表示增益下限、当前增益和增益上限。用户可以通过两种方法改变增益：移动滑块或在当前增益栏中输入新的值。还可以通过改变上、下限来改变增益范围。若用户每点击一次滑尺的左向箭头或右向箭头，当前增益将改变滑尺增益的 10%。点击 Close 按钮关闭对话框。

若希望用于一个矢量增益，考虑使用 Gain 模块（参看“Gain 模块”）。若希望用于一个矩阵增益，请用 Matrix Gain 模块（参看“Matrix Gain 模块”）。

3. 特性

- 数据类型：同 Gain 模块；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：增益；
- 状态数：0；
- 可矢量化。

7.8.19 Sum 模块

1. 功能描述

Sum 模块根据模块输入数对标量和（或）矢量输入，或矢量信号输入的元素进行加运算：

- 若模块的输入不止一个，输出为所有输入对应元素之和。若所有输入均为标量，则输出为一个标量。对于一个有 n 个输入的模块来说，若有一个输入是矢量，则产生的第 i 个输出元素为：

$$y_i = u1_i + u2_i + \cdots + un_i$$

输出只有当所有输入均为标量时才为标量。

- 若模块有一个矢量输入，模块输出为输入元素的标量和：

$$y = \sum u_i$$

注意：当用户将 Simulink 模块库中的 Sum 模块复制到一个模型里时，Simulink 隐含了 Sum 名。

2. 参数和对话框

Sum 模块的参数对话框如图 7.93 所示。

- Icon shape：用户可以在 Icon shape 下拉菜单中选择圆形或方形 Sum 模块。若 Sum 模块有多个输入，最好选择圆形。
- List of signs：List of signs 参数可以是一个常数或是由“＋”、“－”和“|”等符号的组合。对指定的常数，Simulink 将重新画出对应数量和正极性端口的模块。正负号指定对应端口的极性，端口数等于所用符号的数量。

Sum 模块在对应端口的旁边画出正负号并且在重新画出的模块上使端口数与 List of signs 参数的指定数保持一致。若改变符号的数量，端口将相应从图标上添加或删除。若需

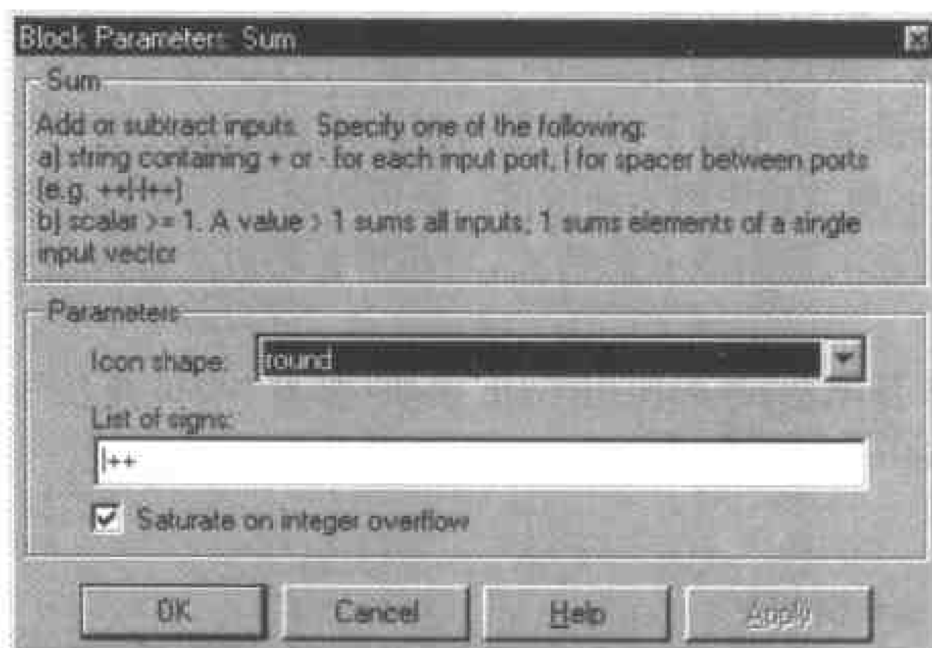


图 7.93　Sum 模块的参数对话框

要，Simulink 重新调整模块的大小以形成所有输入。用户可以在 List of signs 参数的符号之间插入分隔符(|)以调整输入端口的位置。分隔符在端口之间插入一个空格。例如，++|-- 将在第二个+端口和第一个-端口之间插入一个空格。

• Saturate on integer overflow：若选中此选项，Sum 模块输出将在整数溢出时饱和。特别地，若输出是整数型，模块的输出为可能输出的最大值。若此选项未被选中，则 Simulink 将采取 Simulation Parameters 对话框 Diagnostics 选项的 Data overflow(参看第 5 章“Diagnostics 选项”)。

3. 特性

• 数据类型：任意；
• 直通输出；
• 采样时间：从驱动模块继承；
• 标量扩展；
• 状态数：0；
• 可矢量化。

7.8.20　Trigonometric Function 模块

1. 功能描述

Trigonometric Function 模块执行多种常用三角函数。

在 Function 表中可供选择的三角函数：sin，cos，tan，asin，acos，atan，atan2，sinh，cosh 和 tanh。模块输出为函数对输入运算的结果。

函数名显示在模块图标上。Simulink 自动绘制出适当数量的输入端口。模块接受和输出双精度型实或复信号。

用户需要矢量化输出时，请使用本模块而不是 Fcn 模块，因为 Fcn 模块只能产生标量输出。

2. 参数和对话框

Trigonometric Function 模块的参数对话框如图 7.94 所示。

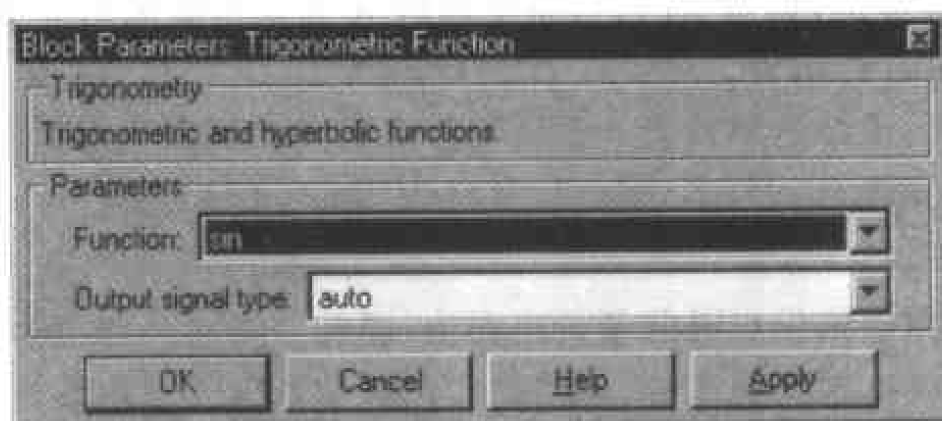


图 7.94 Trigonometric Function 模块的参数对话框

- Function：三角函数。
- Output signal type：要输出的信号(复或实)的数据类型。

3. 特性

- 数据类型：Trigonometric Function 模块接受和输出双精度型实或复信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：输入(当函数要求两个输入时)；
- 可矢量化。

7.9 常用函数和查表库模块

由描述常用函数及查表模块组成。

7.9.1 Fcn 模块 

1. 功能描述

Fcn 模块对输入进行指定的 C 语言格式的数学表达式处理。

(1) 这些表达式可由一个或多个下列成分组成：

- u——模块的输入。如果 u 是一个矢量，u(i)表示矢量的第 i 个元素；u(1)或单独 u 表示第一个元素。
- 常数。
- 算子(+、—、*、/)
- 关系运算子(==、!=、>、<、>=、<=)。若关系成立(True)，表达式返回 1；否则返回 0。
- 逻辑算子(&&、||、!)。若关系成立(True)，表达式返回 1；否则返回 0。
- 括弧。
- MATLAB 数学函数：abs，acos，asin，atan，atan2，ceil，cos，cosh，exp，fabs，floor，

hypot，ln，log，log10，pow，power，rem，sgn，sin，sinh，sqrt，tan 和 tanh。

• 工作空间变量。变量名不能与函数名相同，且必须在 MATLAB 中已经赋值。矩阵或矢量中的元素必须明确指定，如用 A(1,1)而不能用 A 表示矩阵的第一个元素。

(2) 运算优先规则同 C 语言：

- ( )；
- ＋ －(正、负号)；
- pow (求幂)；
- !；
- *、/；
- ＋ －；
- >、<、<=、>=；
- =、! =；
- &&；
- ||。

这里的表达式与 MATLAB 表达式的不同之处在于：这里的表达式不能进行矩阵运算；该模块不支持算子(:)。

模块的输入可以是一个标量或矢量。输出总为标量。如需矢量输出，请使用 Math Function 模块。若输入是矢量，并且函数对输入矢量的单个元素进行运算，则模块仅对第一个矢量元素进行处理。

2. 参数和对话框

Fcn 模块的参数对话框如图 7.95 所示。

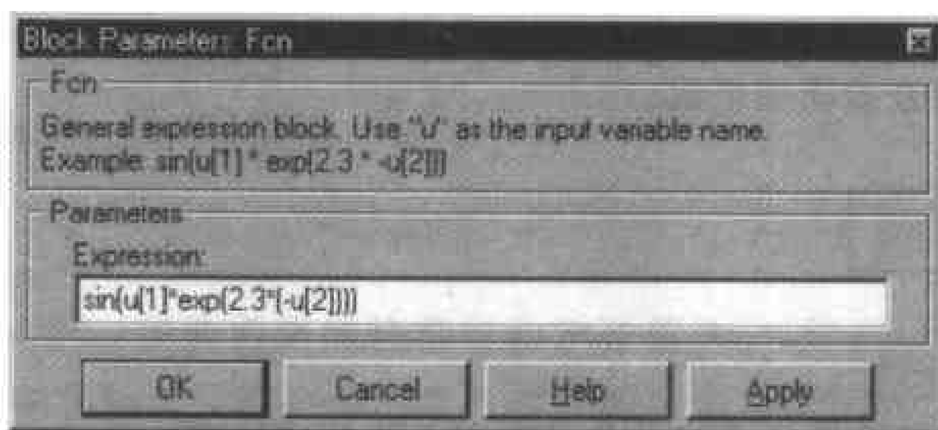


图 7.95　Fcn 模块的参数对话框

• Expression：C 语言格式的表达式，用于对输入信号处理。表达式各个成分列表如图 7.96 所示。表达式必须是完整的。

3. 特性

- 数据类型：Fcn 模块输入和输出双精度型信号；
- 直通输出；
- 采样时间：从驱动模块继承。

7.9.2 Look-Up Table 模块

1. 功能描述

Look Up Table 模块根据模块参数的定义值对输入进行线性插值映射输出。用户通过指定(行或列矢量)Vector of input values 和 Vector of output values 参数定义表。模块经对比输入矢量的各模块输入值产生一个输出。

- 若模块找到与模块输入相匹配的值,在输出矢量的相应元素位输出。
- 若模块找不到与模块输入相匹配的值,则模块在表中两个适当的元素之间进行线性插值运算从而确定输出值。若模块的输入小于输入矢量的第一个元素或大于最后一个元素值,则模块就会用头两个或后两个进行外插计算。

如若需将两个输入映射为一个输出,请参看和使用 Look-Up Table (2-D)模块。

2. 应用举例

若需创建一个步进变化的表,对不同的输出值使用相同的输入。例如,用输入和输出参数值建立输入输出关系图(如图 7.96):

Vector of input values: [−2 −1 −1 0 0 0 1 1 2]

Vector of output values: [−1 −1 −2 −2 1 2 2 1 1]

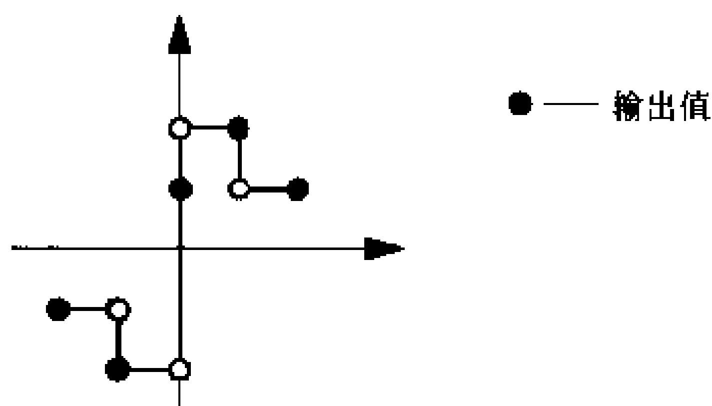


图 7.96 输入输出关系图

此例有三个步进断点,分别在 $u=-1$,$u=0$ 和 $u=+1$ 处。

当一个给定的输入值有两个点时,模块根据如下规则产生输出:

- 当 u 小于 0 时,输出为从原点沿负方向移动所遇到的第一个点的值。本例中,u 等于 -1,y 等于 -2,计实心圆点。
- 当 u 大于 0 时,输出为从原点沿正方向移动所遇到的第一个点的值。本例中,u 等于 1,y 等于 2,计实心圆点。
- 当 u 处于原点并有两个指定 0 输入的输出时,模块的实际输出为它们的平均值。在此例中,在 $u=0$ 和 $y=1$ 没有点,输出将为 0,即在 $u=0$ 处的平均值。若在 0 处有三个点,模块产生与中间值相关的输出。此例中,在原点处,输出为 1。

Look-Up Table 模块的图标显示输入矢量对输出矢量的图形。在模块的对话框上的参数改变后,当用户按下 Apply 或 Close 按钮时,图形就会自动变红。

3. 参数和对话框

Look-Up Table 模块的参数对话框如图 7.97 所示。

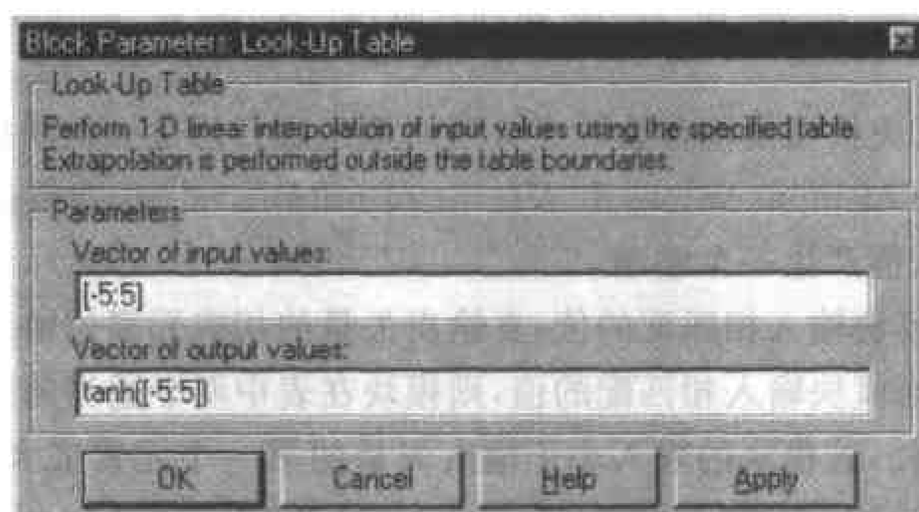


图 7.97　Look-Up Table 模块的参数对话框

- Vector of input values：包含模块的输入值的矢量。该矢量的大小必须与输出矢量相同。输入矢量必须是单调递增的。
- Vector of output values：包含模块的输出值的矢量。该矢量的大小必须与模块输入矢量相同。

4. 特性

- 数据类型：Look-Up Table 模块接受和输出双精度型信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.9.3　Look-Up Table (2-D)模块

1. 功能描述

Look－Up Table(2-D)模块根据模块参数的定义值表对输入进行线性插值映射输出。用户将所有可能的输出值定义为 Table 参数。行与列对应 Row 和 Column 参数。模块用 Row 和 Column 参数经与输入比较后产生输出。第一个输入被看作是行；第二个输入被看作是列，如图 7.98 所示。

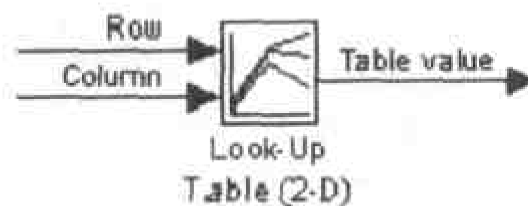


图 7.98　Look-Up Table(2－D)模块输入示意

模块产生的输出基于如下输入值：

- 若输入与行和列参数相匹配，则输出为行和列交点的列表值。
- 若输入与行和列参数不匹配，则模块对适当表值进行线性插值后输出。若两者之一或两个模块输入小于第一个或大于最后一行或列参数值，模块将从头两个或后两个点进行外插计算。

如若 Row 或 Column 参数有相同值，则模块采样 Look－Up Table 模块所描述的方法选择一个值。

Look Up Table 模块允许用户将一个单输入值映射为一个输出值的矢量。参看"Look-Up Table 模块"。

2. 应用举例

设模块参数定义为：

行:[1 2]

列:[3 4]

表:[10 20; 30 40]

图 7.99 说明在行和列参数相匹配的模块输入的交点处输出一个值。第一个输入是 1，第二个输入是 4。它们选择了第一行(行参数值为 1)和第二列(列参数值为 4)的交点处的值。

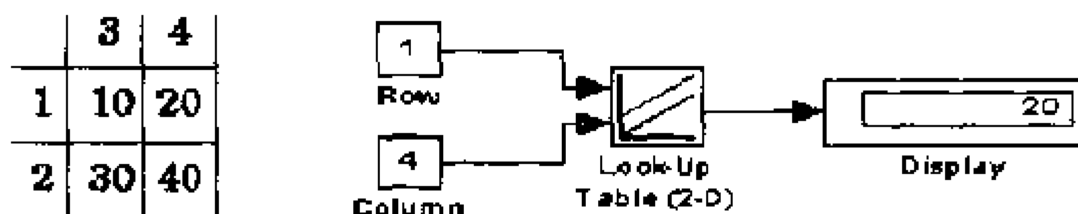


图 7.99 Look-Up Table(2-D)模块输入/输出示意(1)

在图 7.100 中，第一个输入是 1.7；第二个是 3.4。它们使模块在行和列值之间进行插值计算，如左表所示。交点处的 28 即为输出值。

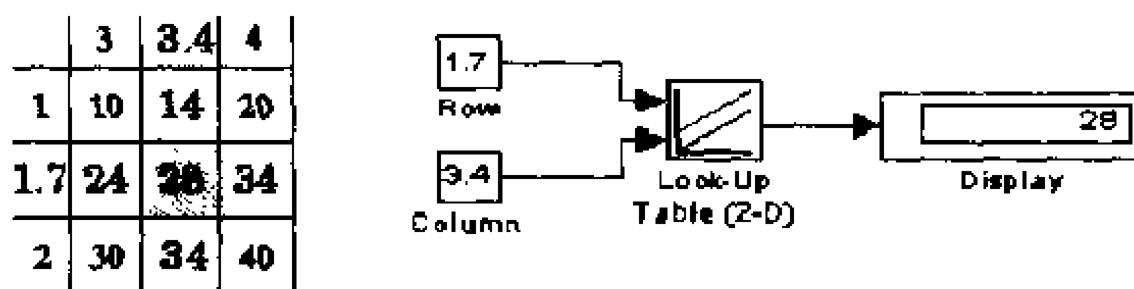


图 7.100 Look-Up Table(2-D)模块输入/输出示意(2)

3. 参数和对话框

Look-Up Table (2-D)模块的参数对话框如图 7.101 所示。

- **Row**:表的行值，矢量输入。矢量值必须单调递增。
- **Column**:表的列值，矢量输入。矢量值必须单调递增。
- **Table**:输出值表。矩阵大小必须同 Row 和 Column 参数定义的维数匹配。

4. 特性

- 数据类型:Look-Up Table (2-D)模块接受和输出双精度型信号；
- 直通输出；
- 采样时间:从驱动模块继承；
- 标量扩展:一个输入，在其他输入为矢量时；
- 可矢量化。

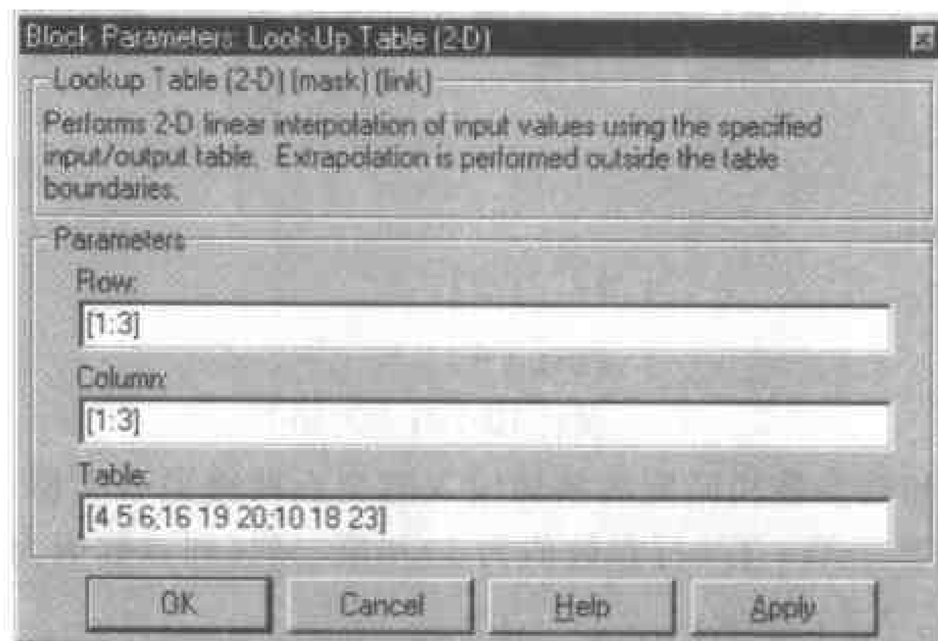


图 7.101 Look-Up Table (2-D)模块的参数对话框

7.9.4 MATLAB Fcn 模块

1. 功能描述

MATLAB Fcn 模块将指定的一个 MATLAB 函数或表达式作为输入。函数的输出必须与模块输出的宽度相匹配，否则会产生错误。

这里给出模块的某些缺省表达式：

```
sin
atan2(u(1), u(2))
u(1)^u(2)
```

编者提示：该模块的执行速度比 Fcn 模块慢，这是因为在每一个集成步内模块都要调用 MATLAB 分析算法。考虑使用定制模块(如 Fcn 模块或 Math Function 模块)，也可以编写函数的 M-file 或 MEX-file S-function，然后通过 S-function 模块对其访问。

2. 参数和对话框

MATLAB Fcn 模块的参数对话框如图 7.102 所示。

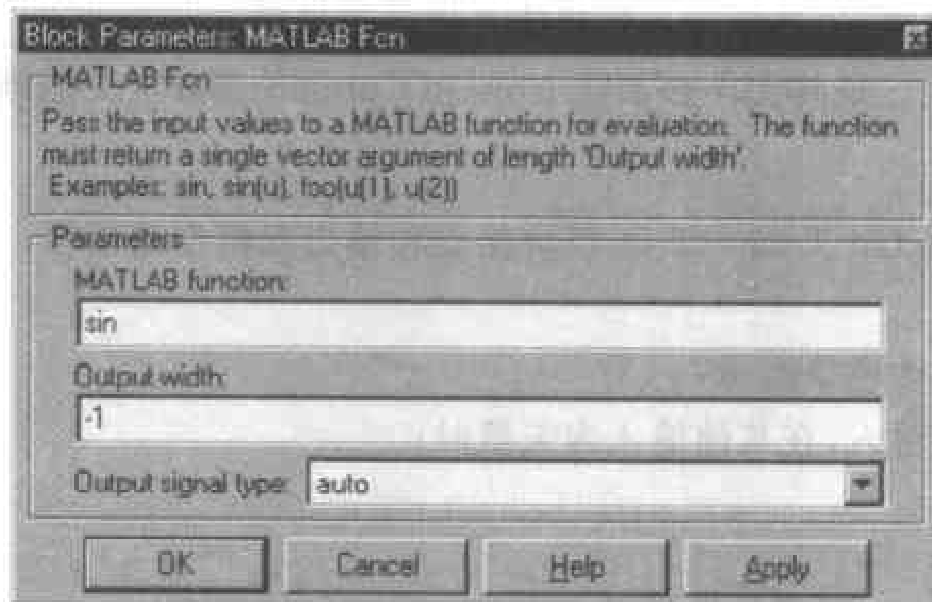


图 7.102 MATLAB Fcn 模块的参数对话框

• MATLAB function：函数或表达式。若用户仅仅指定了一个函数，则括号中没有必要包含输入自变量。

• Output width：输出宽度。若输出宽度需与输入相同，请指定－1，否则用户必须指定恰当的宽度，以免出错。

• Output signal type：对话框允许用户选择"实(real)"、"复(complex)"或"自动(auto)"为 MATLAB Fcn 的输出信号的数据类型。"自动"意味着模块输出信号的数据类型与输入相同。若模块没有输入，将把输出设置为与输出相连接的端口的数据类型。

3. 特性

• 数据类型：MATLAB Fcn 模块接受一个双精度型复或实信号输入，并根据 Output signal type 参数的设置产生双精度型实或复信号输出；

• 直通输出；

• 采样时间：从驱动模块继承；

• 可矢量化。

7.9.5 S-Function 模块 [system]

1. 功能描述

S-Function 模块提供一种调用模块图中的 S-functions(系统函数)的途径。由 S-function name 参数命名的 S-function 可以是一个已写为 S-function 的 M-file 或 MEX-file。

S-Function 模块允许附加参数直接传递给命名的 S-function。函数参数可被指定为 MATLAB 表达式或用逗号分隔的变量。例如，

A, B, C, D, [eye(2,2);zeros(2,2)]

编者提示：虽然个别参数可以用方括号括起来，但参数序列不能。

S-Function 模块显示被指定的 S-function 名，并且总是显示一个输入端口和一个输出端口，而不论包含子系统的输入输出数。

当 S-function 包含了多于一个输入或输出时，将采用行矢量。输入矢量的宽度必须与包含在 S-function 中的输入数量相匹配。模块将输入矢量的第一个元素对准 S-function 的第一个输入，第二个元素对准第二个输入，以此类推。同样地，输出矢量的宽度必须与 S-function 输出数量相匹配。

而模块的数据类型则取决于 S-function 模块的执行过程。

2. 参数和对话框

S-Function 模块的参数对话框如图 7.103 所示。

• S-function name：S-function 名。

• S-function parameters：附加 S-function 参数。

3. 特性

其特性取决于 S-function。

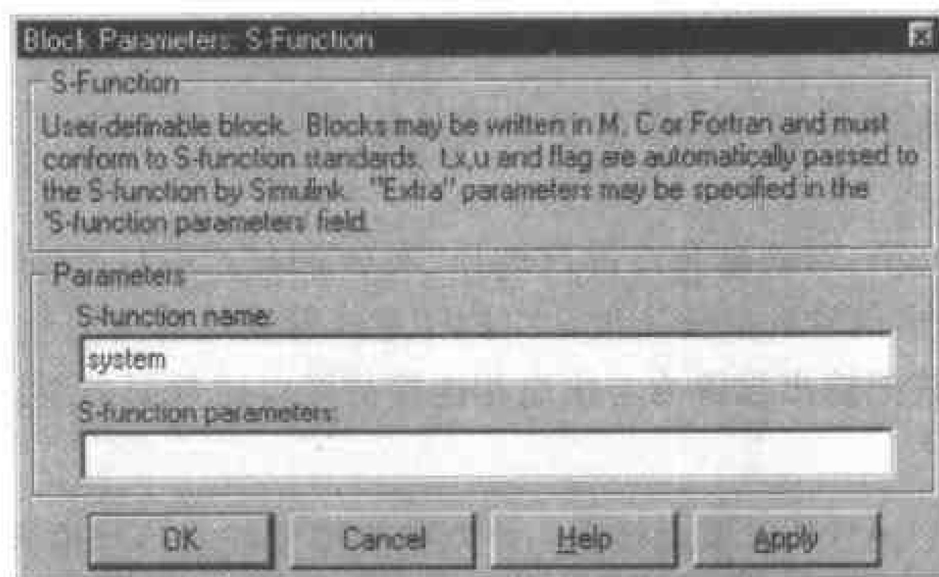


图 7.103 S-function 模块的参数对话框

7.10 非线性系统库模块

由描述非线性函数和非线性系统模块组成。

7.10.1 Backlash 模块

1. 功能描述

该模块可实现输入和输出变化相同的系统。然而，当输入改变其方向时，输入的初始变化对输出没有影响。在系统中，这个四边形的区域称为回差或死区(deadband)。此死区的中心就是输出信号的原点。如图 7.104 示意该模块的初始状态，默认值时死区宽度为 1，对应输出为 0。



图 7.104 关于死区的说明

存在回差的系统有可能是下列三种之一：

- 分离模式——输入信号不控制输出，输出保持为常数。
- 正向工作模式——输入以正斜率上升，而输出等于输入减去死区宽度的一半。
- 负向工作模式——输入以负斜率下降，而输出则为输入加上死区宽度的一半。

如果初始输入落在死区之外，Initial output 参数值将决定模块是正向工作还是负向工作，并且决定在仿真开始时的输出是输入加还是减死区宽度的一半。

例如，Backlash 模块可以用于模拟两个齿轮的啮合。输入和输出皆为一端有齿轮的轴，且输出轴由输入轴驱动。齿轮的齿之间的间隙就会产生回差。间隙的宽度就是 Deadband width 参数。如果系统初始时处于分离模式，则输出(被动齿轮)就由 Initial output 参数的值决定。

2. 应用举例

图 7.105 至图 7.107 说明了当初始输入落在死区中时模块的工作情况。

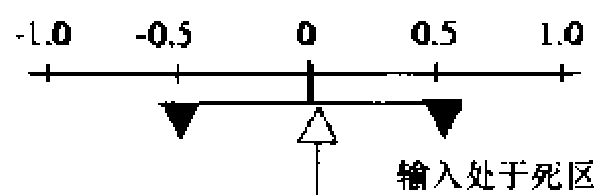


图 7.105　当系统为分离模式(默认值不变)时输入和输出的关系

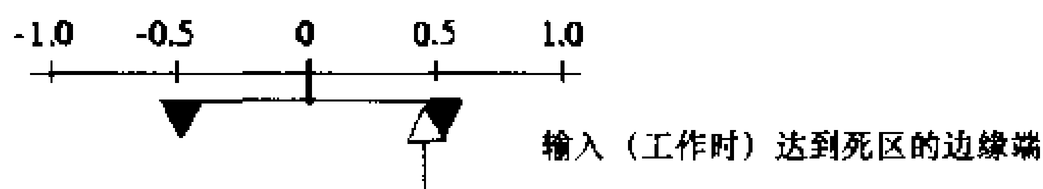


图 7.106　当输入到达死区的边缘时，输出仍然保持原有的值

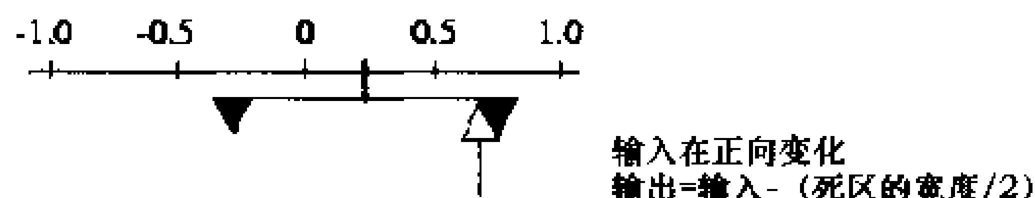


图 7.107　当输入和输出同时作用时，输入的变化对输出的影响

如果输入改变方向，它将脱离对输出的控制。输出将保持不变，直到输入到达死区的另一边缘端或是再次改变方向，工作在原来的死区边缘端。如此这般，输入的变化引起输出等同的变化。

例如，设死区的宽度为 2，初始输出为 5，则在仿真开始时，输出 y：

- 如果输入 u 在 4 和 6 之间，则为 5
- 如果 $u < 4$，则为 $u + 1$
- 如果 $u > 6$，则为 $u - 1$

图 7.108 的采样模型和图 7.109 的图表明了正弦波通过 Backlash 模块后的效果。

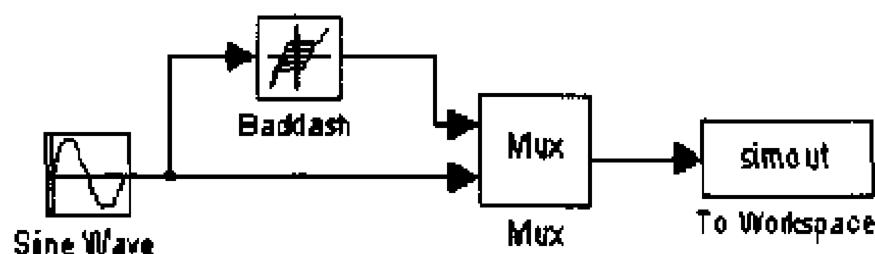


图 7.108　一个采样模型

Backlash 模块的参数是未改变的。默认值时，死区的宽度为 1，输出为 0。

编者提示：在图 7.109 所绘出的输出中，Backlash 模块的输出在输入到达死区边缘端(此时，为 0.5)之前皆为 0。此刻，输入和输出都在变化，而且输出变化与输入相同，直到输入为 1.0 时改变方向为止。当输入为 0 时，它将把输出置于死区的另一端。

3. 参数和对话框

Backlash 模块的参数对话框如图 7.110 所示。

- Deadband width：死区宽度，默认值为 1。

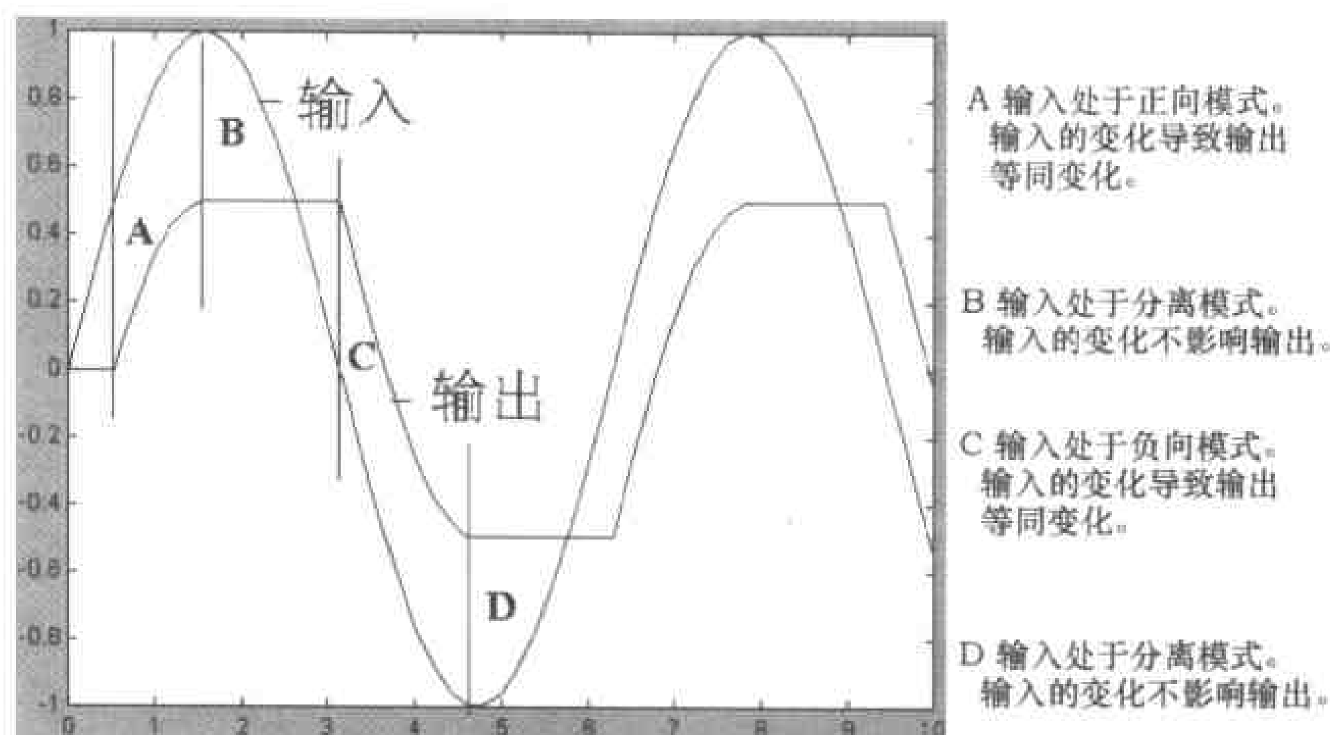


图 7.109 正弦波通过 Backlash 模块后的效果

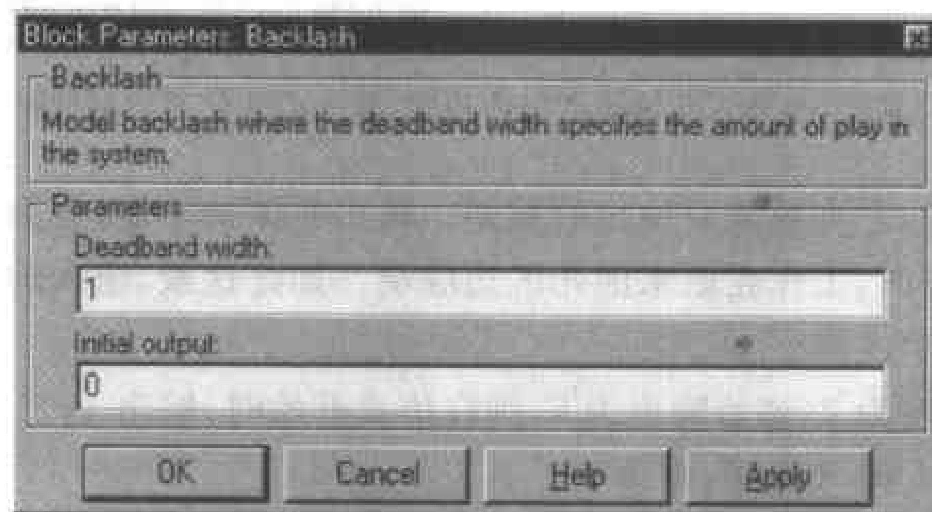


图 7.110 Backlash 模块的参数对话框

- Initial output:初始输出值,默认值为 0。

4. 特性

- 数据类型:Backlash 模块输入和输出皆为双精度实数;
- 直通输出;
- 采样时间:从驱动模块继承;
- 标量扩展;
- 可矢量化;
- 过零检测:决策通过下、上甄别阈。

7.10.2 Coulomb and Viscous Friction 模块

1. 功能描述

Coulomb and Viscous Friction 模块建立库仑(静)力和粘滞(动)力模型。该模块建立的是在零点不连续而其余点线性的增益模型。偏置对应库仑力;增益对应粘滞力。该模块由如

下的函数式表示：

$$y = \mathrm{sign}(u) * (\mathrm{Gain} * \mathrm{abs}(u) + \mathrm{Offset})$$

y 代表输出；u 代表输入；Gain 和 Offset 为模块参数。

模块有一个输入端口和一个输出端口。

2. 参数和对话框

Coulomb and Viscous Friction 模块的参数对话框如图 7.111 所示。

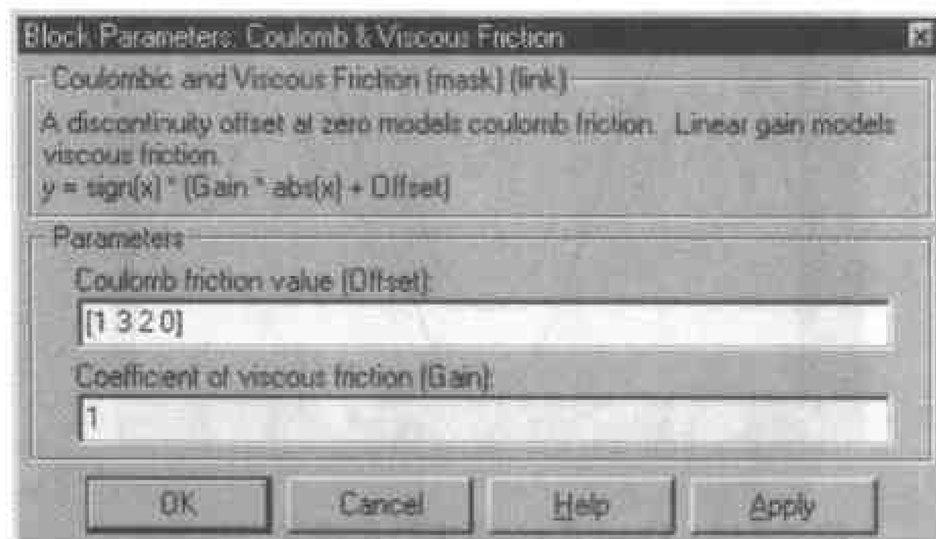


图 7.111　Coulomb and Viscous Friction 模块的参数对话框

- Coulomb friction value：偏置，适用于所有输入。默认值为[1 3 2 0]。
- Coefficient of viscous friction：在非零输入点的信号增益。默认值为 1。

3. 特性

- 数据类型：Coulomb and Viscous Friction 模块输入和输出双精度型实信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 可矢量化；
- 过零检测：在克服静力点上。

7.10.3 Dead Zone 模块

1. 功能描述

Dead Zone 模块产生在指定范围(称为截止区)内的零输出。用 Start of dead zone 和 End of dead zone 参数指定截止区的下限值和上限值。模块的输出取决于输入和截止区的大小：

- 若输入落在截止区域内(大于下限值并小于上限值)，则输出为 0；
- 若输入大于或等于上限值，则输出等于输入减去上限值；
- 若输入小于或等于下限值，则输出等于输入减去下限值。

2. 应用举例

如图 7.112 所示的模型，其下、上限值分别为 −0.5 和 +0.5，并以正弦波作为输入。

图 7.113 说明了 Dead Zone 模块对正弦波的影响。当输入在 −0.5 和 0.5 之间时，输出为 0。

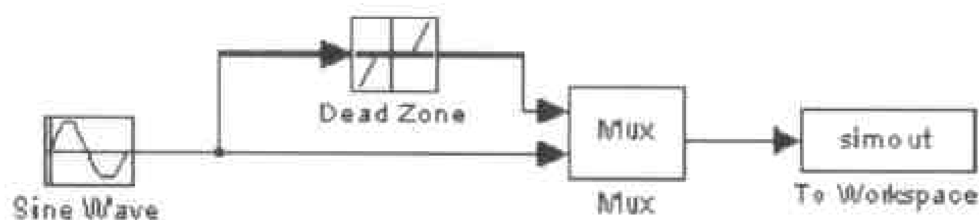


图 7.112　Dead Zone 模块应用举例

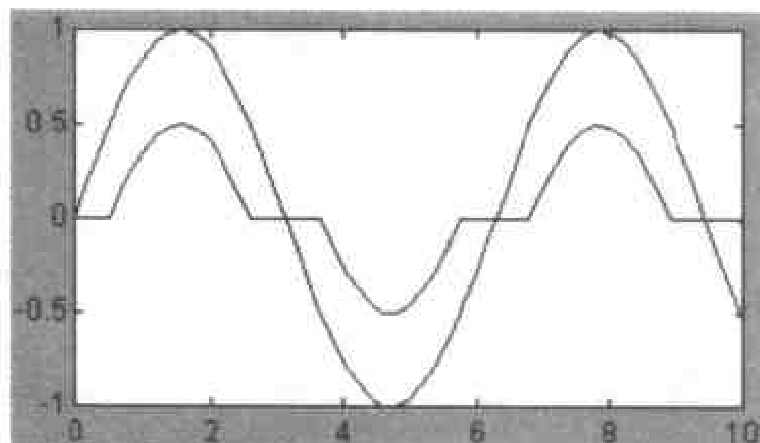


图 7.113　Dead Zone 模块对正弦波形的影响

3. 参数和对话框

Dead Zone 模块的参数对话框如图 7.114 所示。

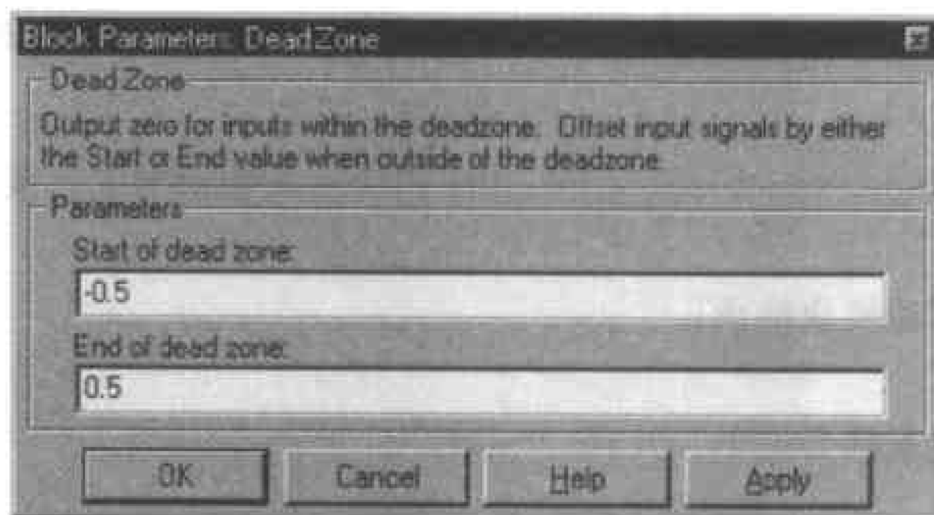
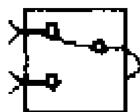


图 7.114　Dead Zone 模块的参数对话框

- Start ofdead zone：截止区的下限。默认值为－0.5。
- End ofdead zone：截止区的上限。默认值为 0.5。

4. 特性

- 数据类型：Dead Zone 模块输入和输出双精度型实信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：参数；
- 可矢量化；
- 过零检测：探测到达限值的时间。

7.10.4　Manual Switch 模块 

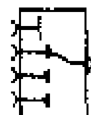
1. 功能描述

Manual Switch 模块相当于一个双态的拨动开关，它选择两个输入之一作输出。双击模块图标就可在两输入间转换。所选输入传送输出，而未选之输入则被禁止。用户可以在仿真之前设置开关，也可在仿真执行期间通过人机对话控制信号流程时将其抛弃。Manual Switch 模块在模型保存后可保留当前状态。

编者提示：模块要求输入应具有相同的数据类型。

2. 特性

- 直通输出；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.10.5　Multiport Switch 模块 

1. 功能描述

从输入之间选择。第一个输入作为控制输入，其他输入为数据输入。控制输入的值决定了哪一个数据输入通过模块输出。

若控制输入是非整数，Multiport Switch 将其取整（取最接近的整数）并提示警告。若取整后的输入值小于 1 或大于总输入端数，模块会给出溢出警告。一般地，模块将把取整后的控制输入所对应的数据输入输出。表 7.15 归纳了 Multiport Switch 模块的性能。

表 7.15　模块性能

（取整）控制输入	输出
小于 1	溢出
1	第一输入
2	第二输出
其余	其余
大于总输入端数	溢出

数据输入可以是标量或矢量。控制输入可以是一个标量或矢量。模块输出由下列规则确定：

- 若输入均为标量，则输出为一个标量。
- 若模块的数据输入超过一个，而且至少有一个是矢量，则输出为一个矢量。任何标量输入均扩展为矢量。

• 若模块仅有一个数据输入并且是一个矢量，则模块输出由取整控制输入所对应之矢量元素组成。

Multiport Switch 模块的控制输入接受除布尔型以外的任意数据类型的实信号。模块的数据输入接受任意数据类型的实或复值输入。所有数据输入必须具有相同的数据和数值类型。

2. 参数和对话框

Multi port Switch 模块的参数对话框如图 7.115 所示。

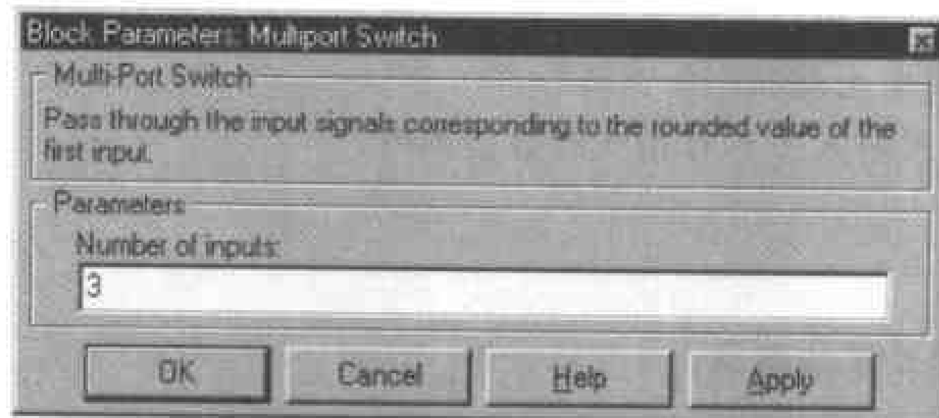


图 7.115　Multi port Switch 模块的参数对话框

• Number of inputs：模块的数据输入数。

3. 特性

• 直通输出；
• 采样时间：从驱动模块继承；
• 标量扩展；
• 可矢量化。

7.10.6　Quantizer 模块

1. 功能描述

量化输入信号。将平滑的输入信号变为阶梯状输出。模块接受和输出双精度型信号。输出计算采用四舍五入法，产生与零点对称的输出。

$$y = q * \mathrm{round}(u/q)$$

其中 y 为输出，u 为输入，q 为 Quantization interval 参数。

2. 参数和对话框

Quantizer 模块的参数对话框如图 7.116 所示。

• Quantization interval：量化输出的时间间隔。Quantizer 模块的输出允许值为 $n*q$，其中 n 是一个整数，q 为 Quantization interval 参数。默认值为 0.5。

3. 特性

• 直通输出；
• 采样时间：从驱动模块继承；
• 标量扩展：参数；

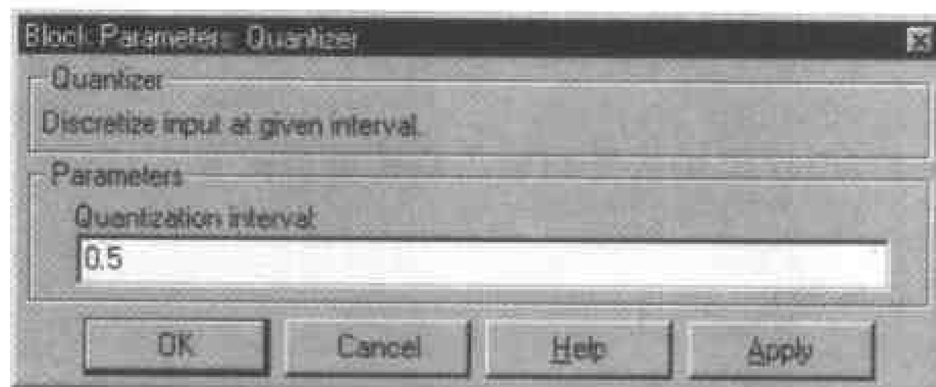


图 7.116 Quantizer 模块的参数对话框

- 可矢量化。

7.10.7 Rate Limiter 模块

1. 功能描述

Rate Limiter 模块限定通过模块的信号的一阶导数，以使输出的变化不超过指定界限。导数根据以下方程计算得出：

$$\text{rate} = \frac{u(i) - y(i-1)}{t(i) - t(i-1)}$$

$u(i)$ 和 $t(i)$ 为当前模块的输入和时间，$y(i-1)$ 和 $t(i-1)$ 为前一时间步的输出和时间。输出通过将 rate 与 Rising slew rate 和 Falling slew rate 参数比较得出

- 若 rate 大于 Rising slew rate 参数（R），输出计算为：

$$y(i) = \Delta t \cdot R + y(i-1)$$

- 若 rate 小于 Falling slew rate 参数（F），输出计算为：

$$y(i) = \Delta t \cdot F + y(i-1)$$

- 若 rate 在 R 与 F 之间，输出的变化与输入的变化相同。

$$y(i) = u(i)$$

2. 参数和对话框

Rate Limiter 模块的参数对话框如图 7.117 所示。

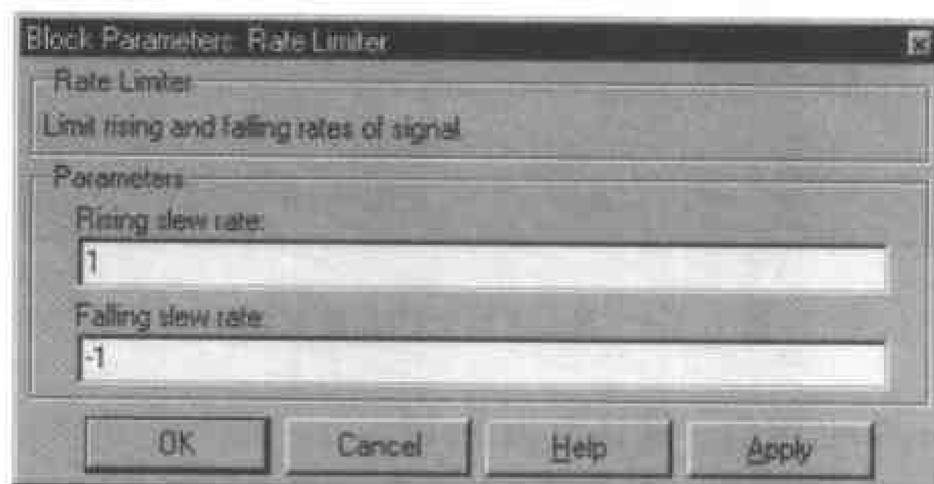


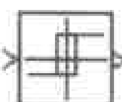
图 7.117 Rate Limiter 模块的参数对话框

- Rising slew rate：一个递增输入信号的导数极限。

- Falling slew rate：一个递减输入信号的导数极限。

3. 特性

- 数据类型：Rate Limiter 模块接受和输出双精度型信号；
- 直通输出；
- 采样时间：连续或离散；
- 标量扩展：参数；
- 可矢量化；
- 过零检测。

7.10.8 Relay 模块 

1. 功能描述

Relay 模块的输出为两个值之一。当模块状态为“on”，此状态一直保持到输入下降到比 Switch off point 参数值小时。若为“off”，此状态一直保持到输入超过 Switch on point 参数值时。模块接受一个输入，产生一个输出。

指定 Switch on point 值大于 Switch off point 值模拟滞后。但若指定的值相等，模块模拟一个以指定值为阈值的开关。Switch on point 值必须大于或等于 Switch off point 值。

2. 参数和对话框

Relay 模块的参数对话框如图 7.118 所示。

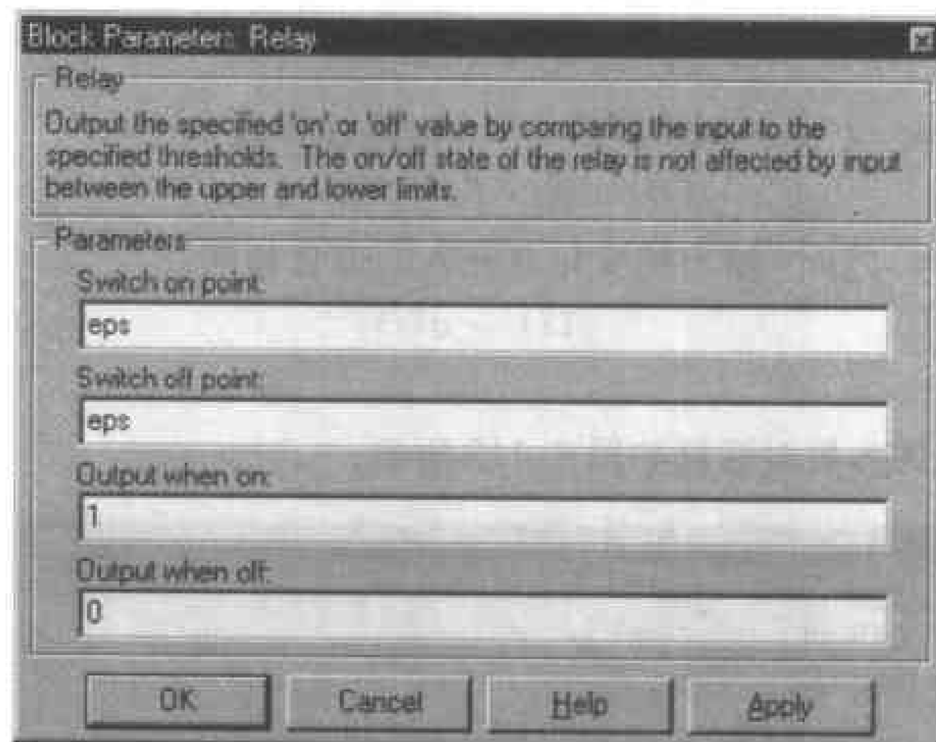


图 7.118 Relay 模块的参数对话框

- Switch on point：“on”阈值。默认值为“eps”。
- Switch off point：“off”阈值。默认值为“eps”。
- Output when on：“on”时的输出。默认值为 1。
- Output when off：“off”时的输出。默认值为 0。

3. 特性

- 数据类型：双精度型实数；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展；
- 可矢量化；
- 过零检测：探测开关“on”，“off”点。

7.10.9 Saturation 模块

1. 功能描述

Saturation 模块对一个信号限定上、下限。当输入在由 Lower limit 和 Upper limit 参数指定的范围内时，输入信号无变化输出。若输入信号超出范围，则信号就会被限幅（值为上限或下限）。若这两个参数的设置值相等，模块就输出该值。模块只接受和输出双精度型实信号。

2. 参数和对话框

Satuation 模块的参数对话框如图 7.119 所示。

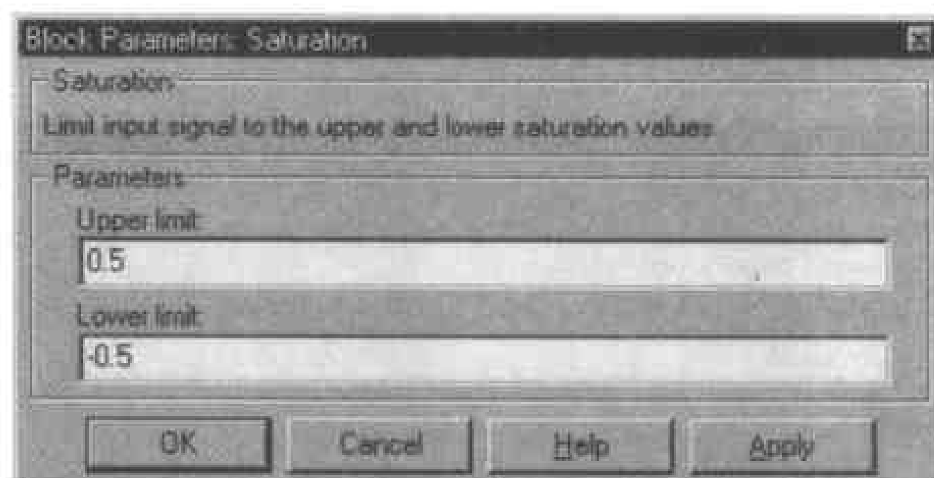
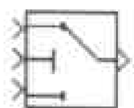


图 7.119　Satuation 模块的参数对话框

- Upper limit：限定输入信号的上限。若输入信号大于此值，模块设置输出为此值。
- Lower limit：限定输入信号的下限。若输入信号小于此值，模块设置输出为此值。

3. 特性

- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：参数和输入；
- 可矢量化；
- 过零检测：探测信号到达或离开限定值的时间。

7.10.10 Switch 模块 

1. 功能描述

Switch 模块根据第三个称之为控制输入的输入，选择两个输入之一作为模块的输出。若控制输入(第二个输入端口)的信号大于或等于 Threshold(阈值)参数值，模块将第一个输入作为输出；否则将第三个输入作为输出。图 7.120 给出本模块的使用方法。

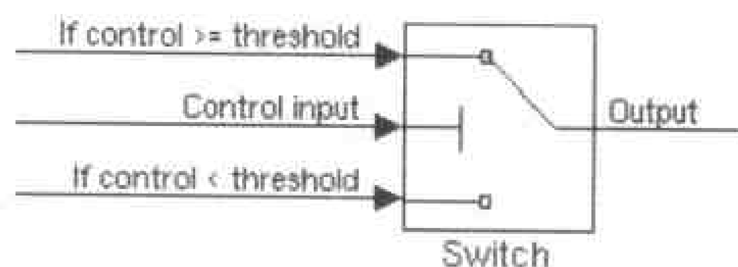


图 7.120　Switch 模块的使用方法

若需用逻辑输入(即 0 或 1 电平)作为控制，将阈值设置为 0.5。

Switch 模块接受作为选择输入(第一和第三输入)的任意数据型的实或复信号。模块输出信号的数据型与所选输入信号的数据型相同。控制输入(阈值)必须为布尔或双精度型。

2. 参数和对话框

Switch 模块的参数对话框如图 7.121 所示。

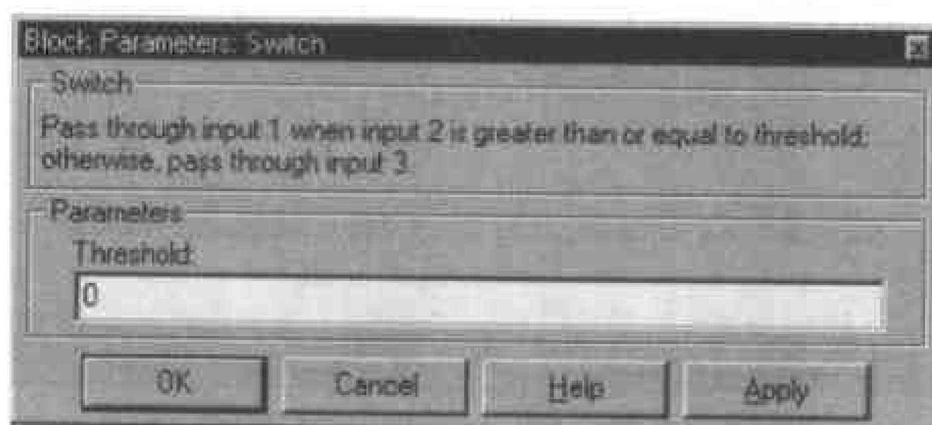


图 7.121　Switch 模块的参数对话框

- Threshold：控制输入(第二个输入)的值。用户可以将该参数指定为一个标量或宽度等于输入矢量的矢量。

3. 特性

- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展；
- 可矢量化；
- 过零检测：探测转换条件发生的时间。

7.11 信号与系统库模块

7.11.1 Bus Selector 模块

1. 功能描述

Bus Selector 模块接受来自 Mux 模块或其他输入 Bus Selector 模块的信号。本模块只有一个输入端口。输出端口的数量取决于 Muxed output 复选框的状态。如果选中 Muxed output，则信号在仅有的一个输出端口被混合；否则，对一个输入就有一个输出。

Simulink 隐藏了从 Simulink 模块库拷贝到模型中的 bus selector 模块名。

2. 参数和对话框

Bus Selector 模块的参数对话框如图 7.122 所示。

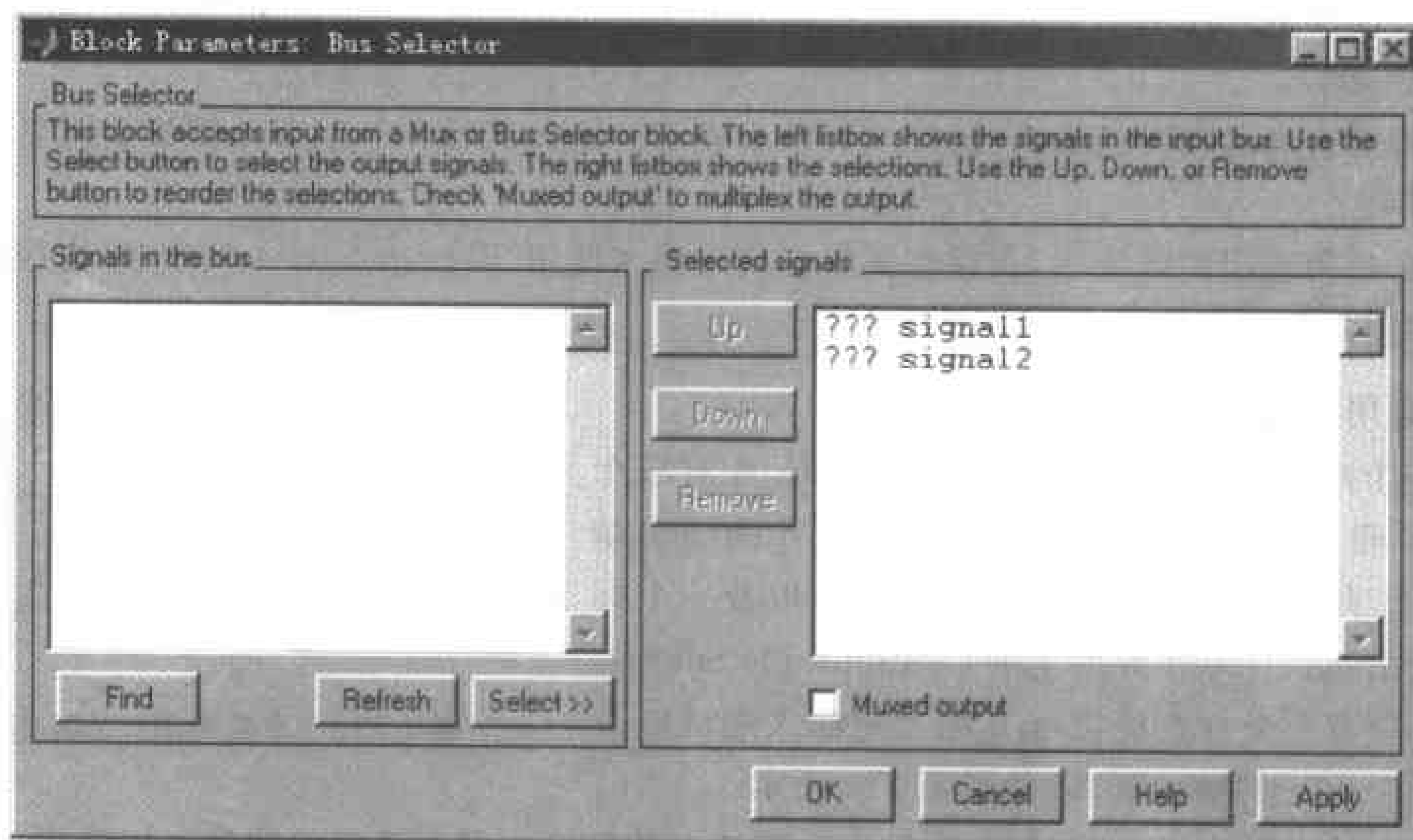


图 7.122 Bus Selector 模块的参数对话框

- Signals in the bus：此列表框显示在输入母线上的信号。用 Select>>选择 Signals in the bus 列表栏上的信号输出。
- Selected signals：此列表框显示输出信号。可以通过 Up，Down 和 Remove 键指定信号。当信号的指定改变后，端口保持原有的连通状态。

如果在 Selected signals 列表选择的输出信号不是 Bus Selector 模块的输入，则信号前冠以“???”。

如果没有选中 Muxed output 复选框，在输出端口的信号都会被自动地打上标记。如果试图更改标记，就会有错误信息提示：不可更改与 Bus Selector 模块在线连接的信号标记。

- 数据类型：Bus Selector 模块接受和输出任意类型数据的实数和复数值。

7.11.2　Configurable Subsystem 模块

1. 功能描述

本模块可以模仿任何一个包含在用户定义模块库中的模块。用户通过其模块的对话框指定需要仿真的模块和参数值。本模块简化了设计同类模型的创建过程。假设要创建一个可供选择发动机的汽车模型。首先应建立一个可用于汽车的发动机的模型库；然后将 Configurable Subsystem 模块用于汽车模型，以提供选择发动机。为了建立设计基本汽车的特殊变量，用户需要做的仅仅是通过使用可设置发动机模块的对话框选择发动机的类型。

一个 Configurable Subsystem 模块外部特性的改变取决于它所模仿的模块。在指定任务之前，Configurable Subsystem 模块不模仿任何模块，因而它没有端口，仅显示图标。当选定了一个库和模块，Configurable Subsystem 模块就会显示对应于所选库之输入、输出端口的图标和一组输入和输出。

Simulink 根据如下法则映射库的端口于 Configurable Subsystem 模块：

• 将库中惟一命名的输入/输出端口映射在 Configurable Subsystem 模块上同名的独立输入/输出端口。

• 将所有命名相同的输入/输出端口映射在 Configurable Subsystem 模块上同名的输入/输出端口。

• 在当前选定库模块中，若存在某些应使用而未使用 Terminator/Ground 模块连接的输入/输出端口，则不映射。

2. 应用举例

有一个库，包含两个模块 A 和 B。模块 A 有三个标以 a，b 和 c 的输入端口以及一个标以 d 的输出端口；模块 B 有两个标以 a 和 b 的输入端口以及一个标以 e 的输出端口。基于此库上的 Configurable Subsystem 模块应有三个分别标以 a，b 和 c 的输入端口以及两个标以 d 和 e 的输出端口，如图 7.123 所示。

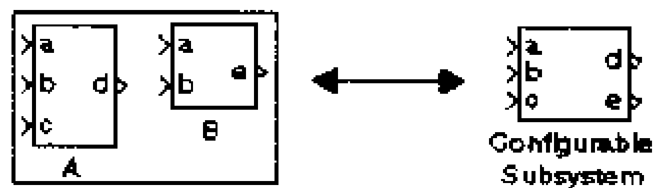


图 7.123　库模块端口的映射

在此例中，在 Configurable Subsystem 模块上的端口 a 和所选库的端口 a 相连接，而不论是哪个被选中的模块。另外，仅当库模块 A 被选中时，Configurable Subsystem 模块上的端口 c 才有效；否则它被终止。

编者提示：Configurable Subsystem 模块对非 I/O 端口的模块不提供端口，如触发加使能子系统的触发和使能端口。因此，不能将 Configurable Subsystem 模块直接用于有这些端口的模块。然而，可以通过将这些模块包装在有和非 I/O 端口相连接的输入或输出端口的子系统模块中来实现。

3. 参数和对话框

Configurable Subsystem 的对话框随 Configurable Subsystem 当前模仿的库以及模仿的模块而改变。对话框中的 Library name 栏初始显示为空（如图 7.124 所示）。

• Library name：Configurable Subsystem 能模仿的模块库的相关路径名，如 simulink/Math。在 Library name 栏中输入希望 Configurable Subsystem 模仿的模块库名。

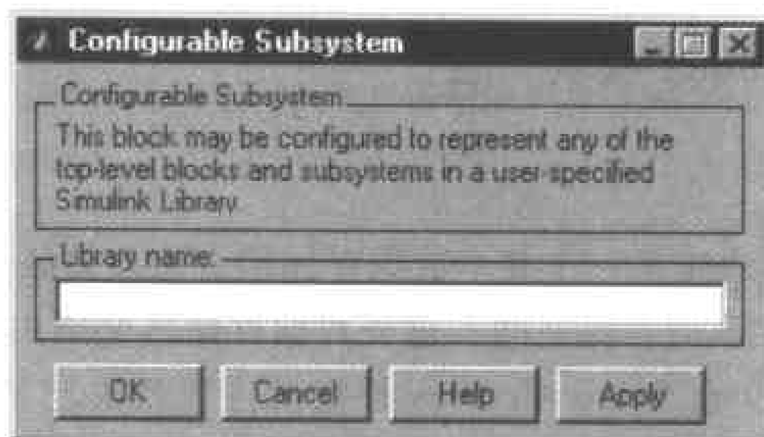


图7.124 Configurable Subsystem 模块的参数对话框

编者提示：

• 不能用对话框更改已由现有 Configurable Subsystem 模块模仿的库。然而，可以通过用 set_param 建模命令设置新库名的模块库参数来改变库。

• 如果在一个库内增加或移动模块，必须重新创建使用该库的 Configurable Subsystem 模块。

当指定该库后，显示新的对话框(如图 7.125 所示)，Library name 栏被新的参数所取代。新的参数包括 Block choice 栏，Open subsystems when selected 栏和被 Configurable Subsystem 当前模仿的模块的参数。

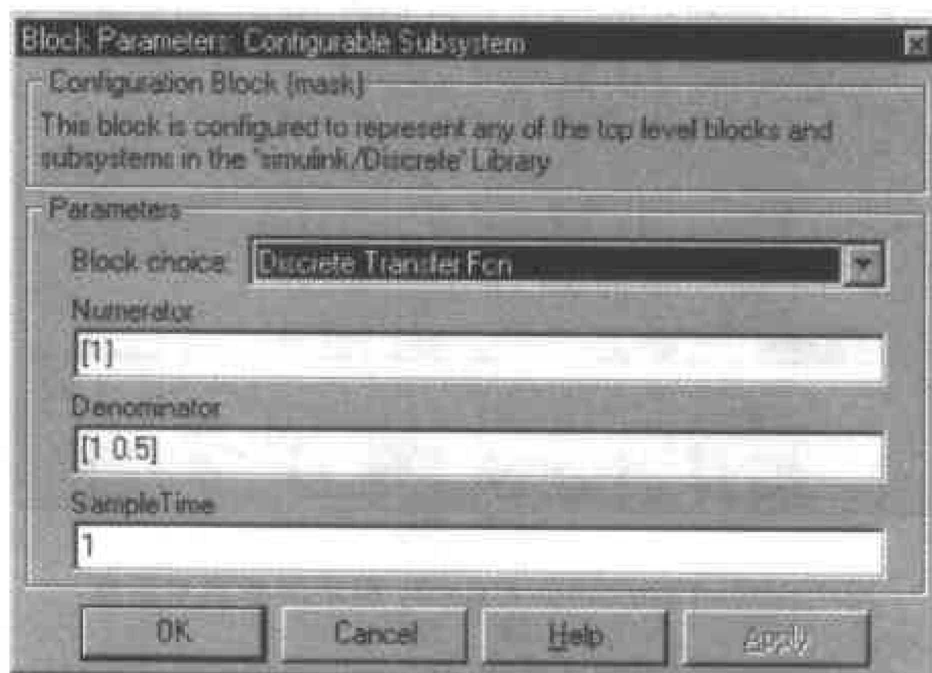


图 7.125 Configurable Subsystem 模块模仿 Simulink Discrete 库模块时的对话框

• Block choice：Configurable Subsystem 模块当前模仿的模块。

• Open subsystems when selected：此选项令 Simulink 打开非封装的模块。此参数仅在被选模块是封装模块时出现。

编者提示：对话框内所显示的其他参数栏是本例中的 Configurable Subsystem 模块所模仿的 Discrete Transfer Function 模块的参数。

4. 特性

Configurable Subsystem 模块的特性同它当前模仿的模块。

7.11.3 Data Store Memory 模块

1. 功能描述

Data Store Memory 模块定义一个共享数据存储区。该存储区是与 Data Store Read 模块和 Data Store Write 模块共享的存储空间。这种数据存储区必须由 Data Store Memory 模块定义。定义数据存储区的 Data Store Memory 模块的位置就决定了 Data Store Read 和 Data Store Write 模块访问此数据存储区的性质：

- 如果 Data Store Memory 模块是在最高一级的系统中，则处于模型中任何位置的 Data Store Read 模块和 Data Store Write 模块都可以访问该数据存储区。
- 如果 Data Store Memory 模块是在子系统中，并且 Data Store Read 和 Data Store Write 模块也位于该子系统或位于子系统的模型分层结构的下级子系统中，也能访问该数据存储区。

通过指定 Initial value 参数的值对数据存储区进行初始化，且定义的值就决定了数据存储区的空间大小。如果 Data Store Write 模块写入的数据量不同，就会产生错误。

2. 参数和对话框

Data Store Memory 模块的参数对话框如图 7.126 所示。

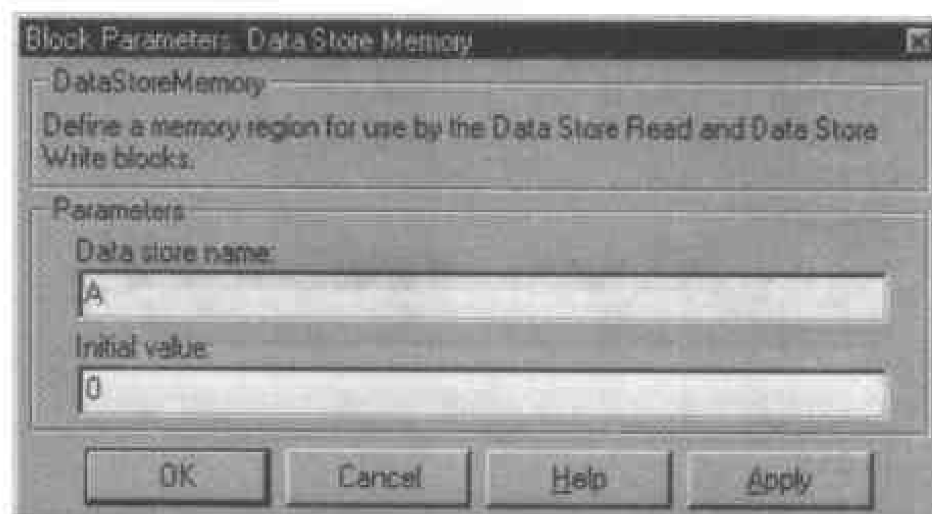


图 7.126 Data Store Memory 模块的参数对话框

- Data store name：正在定义的数据存储区的名。默认值是 A。
- Initial value：数据存储区的初始值。默认值为 0。

3. 特性

- 数据类型：Data Store Memory 模块存储双精度型实信号。
- 可矢量化。

7.11.4 Data Store Read 模块

1. 功能描述

Data Store Read 模块从已定义的数据存储区中读取数据并输出。这些数据是先前由 Data Store Memory 模块初始化并(可能)由 Data Store Write 模块写入数据存储区的数据。

编者提示:多个 Data Store Read 模块可以读取同一个数据存储器里的数据,但受制于 Data Store Memory 模块在系统中的位置。请参看“Data Store Memory 模块”。

2. **参数和对话框**

Data Store Read 模块的参数对话框如图 7.127 所示。

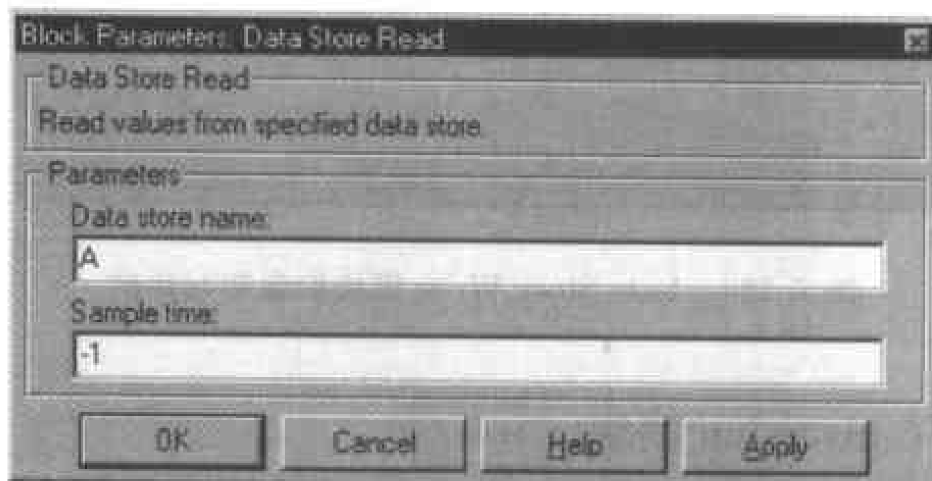


图 7.127 Data Store Read 模块的参数对话框

- Data store name:欲读取数据的数据存储区名。
- Sample time:控制模块读取数据的时间。默认值为-1,表示时间是继承性的。

3. **特性**

- 数据类型:Data Store Read 模块输出双精度型实信号;
- 采样时间:连续或离散;
- 可矢量化。

7.11.5 Data Store Write 模块

1. **功能描述**

将模块的输入数据写入一个已定义的数据存储区。每一次写操作后,原有的数据均被覆盖。而且由 Data Store Write 模块写入的数据量必须与由 Data Store Memory 模块定义的数据存储区一致(数据量及数据类型)。参看“Data Store Memory 模块”。

一般而言,允许多个 Data Store Write 模块对同一个数据存储区进行写操作。然而,如果在同一个仿真步内有两个 Data Store Write 模块试图对相同的存储区进行写操作,后果将难以预料。

2. **参数和对话框**

Data Store Write 模块的参数对话框如图 7.128 所示。

- Data store name:欲写入数据的数据存储区名。
- Sample time:控制模块写入数据的时间。默认值为-1,表示时间是继承性的。

3. **特性**

- 采样时间:连续或离散;
- 可矢量化。



图 7.128　Data Store Write 模块的参数对话框

7.11.6　Data Type Conversion 模块 (double)

1. 功能描述

Data Type Conversion 模块将输入信号转换成由本模块的 Data type 参数所指定的数据类型。输入可以是任何实或复信号。若输入是实信号，输出也将是实信号；若输入为复信号，则输出为复信号。

2. 参数和对话框

Data Type Conversion 模块的参数对话框如图 7.129 所示。

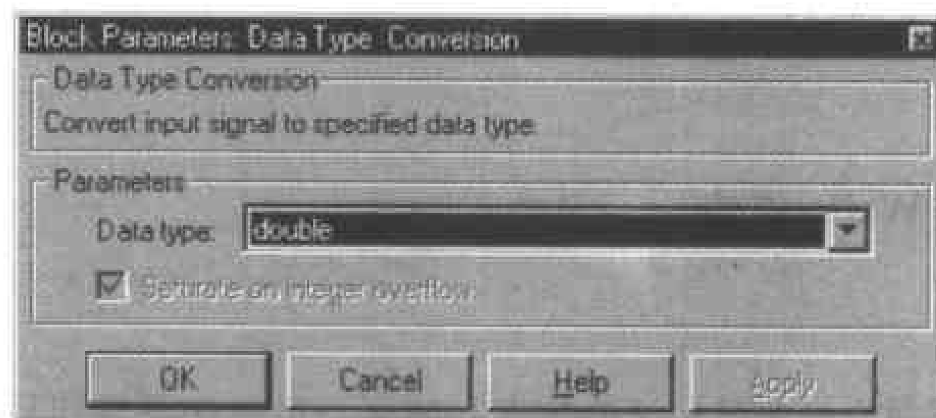


图 7.129　Data Type Conversion 模块的参数对话框

- Data type：指定输入信号将要转换成的数据类型。Auto 选项将输入信号转换成与 Data Type Conversion 模块输出端口相连接的输入端口所要求的类型。
- Saturate on integer overflow：此参数仅在整数型输出时才被激活。若被选中，Data Type Conversion 模块的输出在整数型溢出时饱和。具体来说，如果输出是整数型数据，模块的输出为输出类型或已变换的输出所能描述的最大值，而不论哪一个在绝对意义上是小的。若该选项未被选中，Simulink 采取在 Simulation Parameters 对话框的 Diagnostics 上之 Data overflow 事件选项所指定的动作。请参看第 5 章“Diagnostics 选项”的有关内容。

3. 特性

- 直通输出；
- 采样时间：从驱动模块继承；

- 标量扩展：参数；
- 可矢量化；
- 过零检测：探测到达限值的时间。

7.11.7 Demux 模块

1. 功能描述

Demux 模块将一个矢量输入信号分解成行输出，每一行可包含一个标量或矢量信号。Simulink 通过 Number of outputs 参数决定输出信号的数量和宽度。

(1) 输出标量数量。若 Number of outputs 参数栏包含一个标量值，模块将把输入信号分解成这个数量的信号输出。输出信号的宽度取决于输入矢量和输出的数量：

- 若输入信号的宽度等于输出信号的数量，模块将把输入信号矢量分解成标量信号。在图7.130所示的模型中，Demux 模块将三元素矢量信号分解成三个标量信号。它的 Number of outputs 参数值是 3。

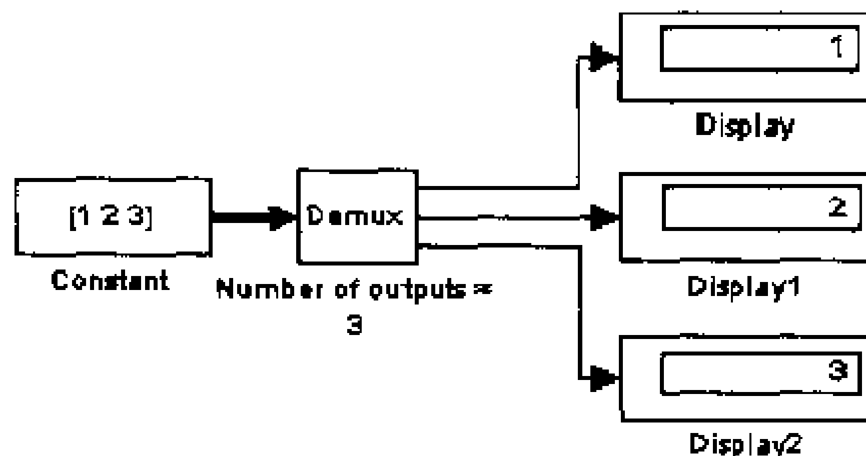


图 7.130 Demux 模块应用举例(1)

- 若输入信号宽度能被输出数量均分，Demux 模块将把输入信号分解成相同宽度的矢量信号输出。在如图 7.131 所示的模型中，Demux 模块将一个 12 元素的矢量信号分解成三个 4 元素宽的矢量信号输出。

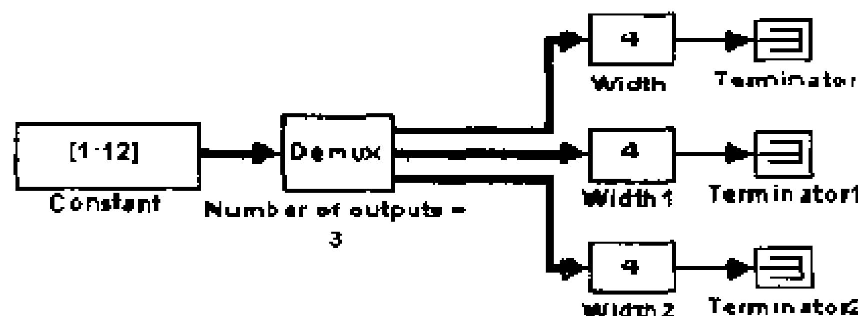


图 7.131 Demux 模块应用举例(2)

- 若输入信号宽度不能被输出的数量均分(输出宽度不同)，模块将把输入信号分成不同宽度的矢量，Simulink 给出提示信息。在图7.132所示的模型中，Demux 模块把一个 4 元素矢量信号分成了三个信号。第一个信号包含了输入信号的头两个元素。

(2) 输出矢量数量。如果 Number of outputs 参数是矢量，输出的数量等于矢量中的元素

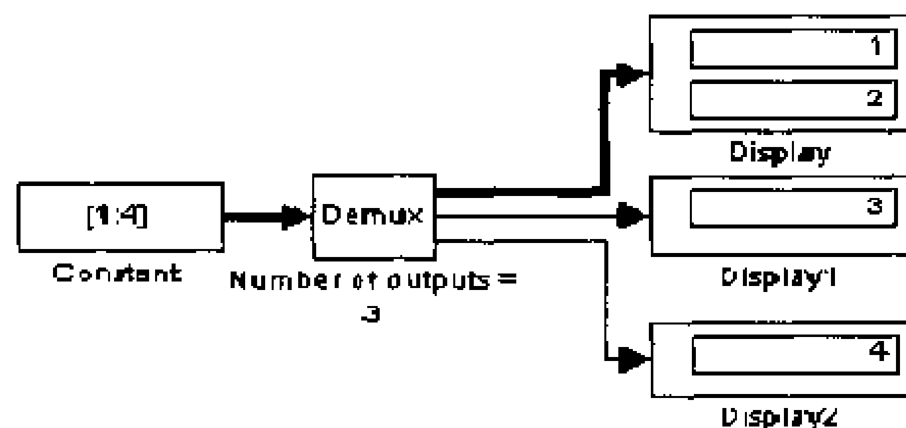


图 7.132 Demux 模块应用举例(3)

数量。输出信号的宽度取决于输入矢量宽度和参数的元素值。用户可以指明输出信号的大小或让 Simulink 决定输出的宽度。

• 如果 Number of outputs 矢量元素都是正值,模块产生为指定宽度的信号。在图7.133所示的模型中,输入信号是宽度为 7 的矢量,Number of outputs 参数为[2 4 1]。

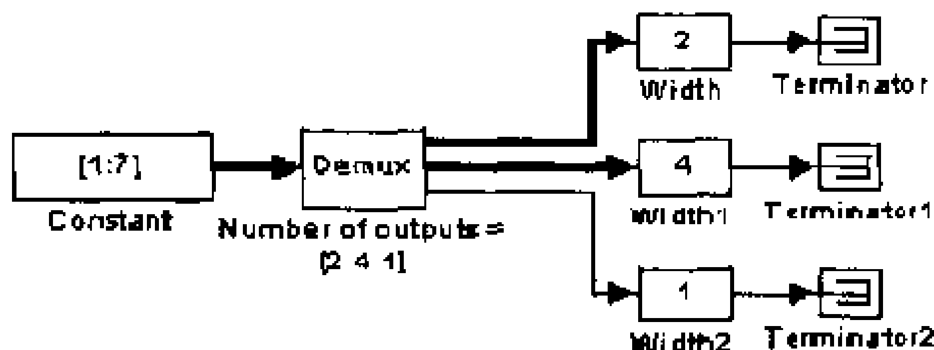


图 7.133 Demux 模块应用举例(4)

• 如果 Number of outputs 矢量元素包括正数和—1 值,模块将给那些正值输出产生指定宽度的输出,并且自动安排为—1 值的输出。

在图 7.134 所示的模型中,输入信号是宽度为 7 的矢量,Number of outputs 参数是[—1 3 —1]。Simulink 产生了一个三元素的矢量信号作为第二输出并且尽可能把剩下的输入均匀地分成两个输出。在此种情况下,4 个元素均匀划分。

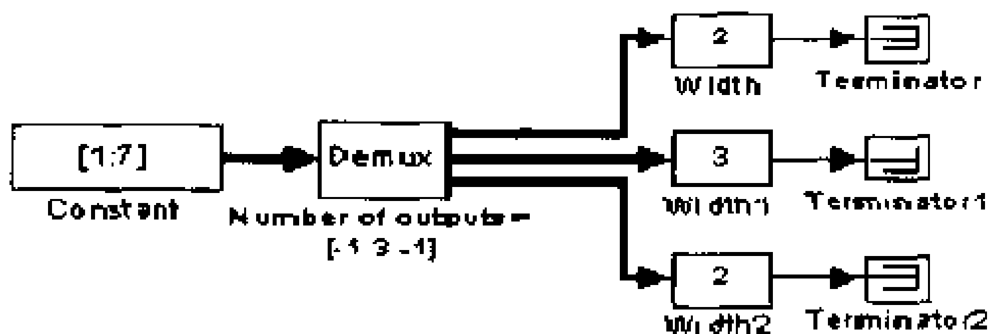


图 7.134 Demux 模块应用举例(5)

而在图 7.135 所示的例子中,Number of outputs 被指定为[—1 4 —1]。这使得 Simulink 产生宽度不同的矢量。

• 如果 Number of outputs 矢量元素全为—1,则输出的数量就等于矢量元素的数量,宽度也自动调整。按此种方式指定参数就如同将参数指定为一个其值等于元素的数量一样。例

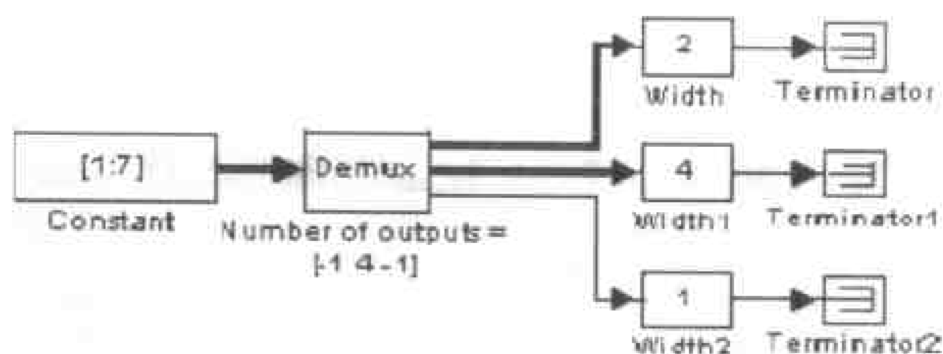


图 7.135　Demux 模块应用举例(6)

如，输入[−1,−1,−1]就等于指定参数值为 3 一样。

Simulink 会在模块上画出指定数量的输出端口，若需要，则调整模块的大小。当端口数量增加或减少时，端口从模块图标底部开始添加或删除。

(3) 用变量设置 Number of Outputs 参数。当用户把 Number of Outputs 参数指定为变量时，如果该变量在 MATLAB 工作空间里不存在，Simulink 将给出错误信息。

当用户将 Simulink 库的模块拷贝到模型中后，Simulink 不显示模块名。

2. 参数和对话框

Demux 模块的参数对话框如图 7.136 所示。

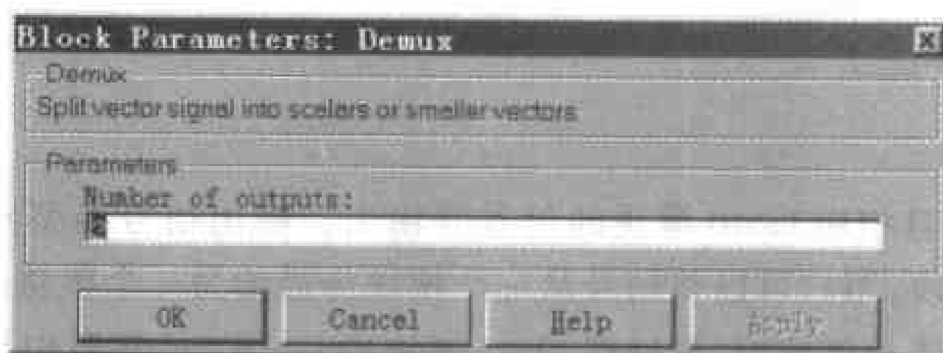


图 7.136　Demux 模块的参数对话框

- **Number of outputs**：输出的数量和宽度。输出宽度的总和必须等于输入行宽。

7.11.8　Enable 模块

1. 功能描述

加上 Enable 模块的子系统就成为“使能(激活)子系统”。只有当进入 Enable 端口的输入大于 0 时，这种子系统才运行。

仿真启动时，Simulink 按照初始条件将包含在使能子系统内的模块初始化。当一个使能子系统被激活而再启动时，States when enabling 参数决定该子系统内模块的状态：

- reset：按照初始条件设置状态；若不知道初始条件，则置 0。
- held：保持原有状态。

用户可以通过 Show output port 复选框选择输出“使能”信号。此选项让系统处理其“使能”信号。一个子系统只允许有一个 Enable 模块。

编者提示：为更好地应用，请参看第 6 章有关“条件执行子系统”的内容。

2. 参数和对话框

Enable Port 模块的参数对话框如图 7.137 所示。

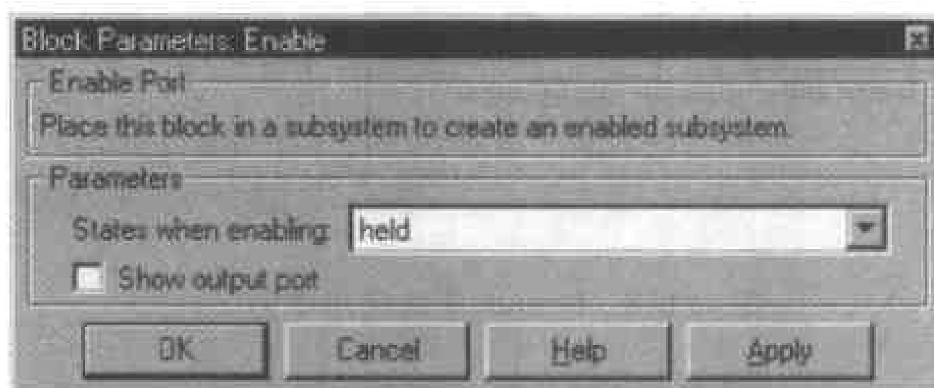


图 7.137 Enable Port 模块的参数对话框

- States when enabling:指定当子系统再次被激活时,处理状态的方式。
- Show output port:若选中,Simulink 给 Enable 模块画一个输出端口并输出使能信号。

3. 特性

- 采样时间:由在使能端口的信号决定;
- 可矢量化。

7.11.9 From 模块

1. 功能描述

From 模块接受来自相应 Goto 模块的信号并输出。输出信号类型同 Goto 模块的输出。通过 From 和 Goto 模块,用户可以将信号从一个模块传递给另一个模块而不需用线连接。为了 From 和 Goto 模块能相伴使用,最好在 Goto 模块的 Goto tag 参数中写上标签。

虽然 Goto 模块可以将信号传递给多个 From 模块,但一个 From 模块只能接受一个来自 Goto 模块的信号。

图 7.138 表明用一个 Goto 模块和一个 From 模块等同于两个模块线连接。在左边的模型中,Block1 将信号传递给 Block2。这相当于在右图中,将 Block1 与 Goto 模块相连,然后通过 From 模块再将信号传递给 Block2。

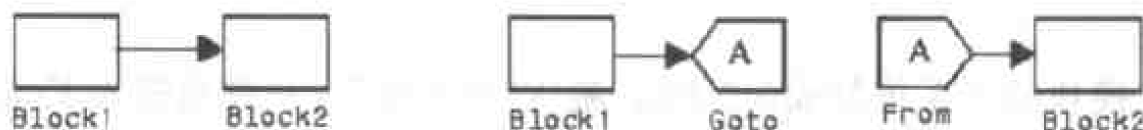


图 7.138 两种等价的模型

Goto 和 From 模块可以在任何一个模型中联合使用。但应注意的是,如果一个模块是在一个条件执行子系统中,则另一个模块必须是在同一个子系统或是在模型层次的下一级子系统中(不在另一个条件执行子系统中)。不过,如果一个 Goto 模块与一个状态数端口相连,则信号可以送至在另一个条件执行子系统中的 From 模块。欲知有关条件执行的详细内容,请参看第 6 章。

一个 Goto 模块标签是否可见,决定了 From 模块能否接收来自 Goto 模块的信号。详细

内容请参看“Goto 模块”及其“Goto Tag Visibility 模块”。模块的图标表明了 Goto 模块标签是否可见：

- 一个局部标签名被括在方括号([])中；
- 一个区域标签名被括在({})中；
- 一个全局标签名出现时无任何标识修饰。

2. 参数和对话框

From 模块的参数对话框如图 7.139 所示。

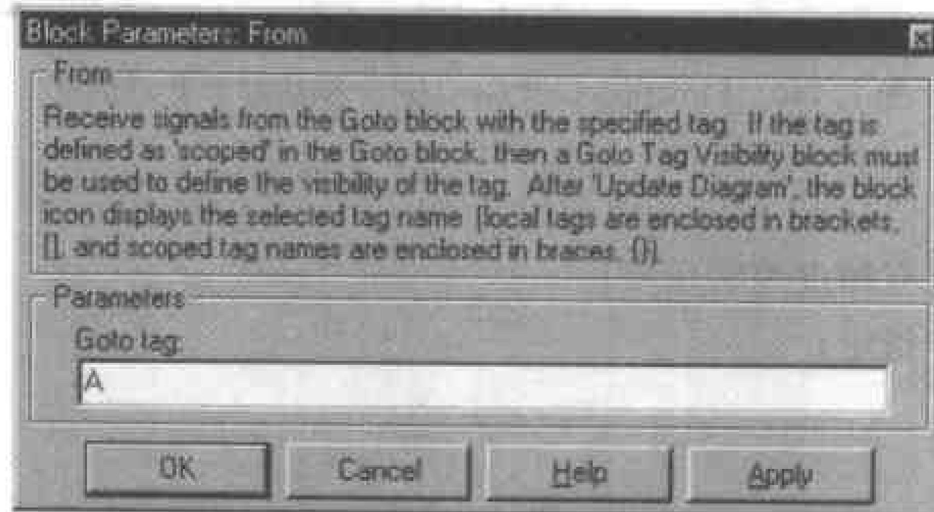


图 7.139　From 模块的参数对话框

- Goto tag：将信号传递给本模块的 Goto 模块标签。

3. 特性

- 采样时间：从驱动 Goto 模块的模块继承；
- 可矢量化。

7.11.10　Function-Call Generator 模块

1. 功能描述

Function－Call Generator 模块以模块的 Sample time 参数指定的采样时间执行一个函数调用子系统。

例如，一个状态数流程可设定为一个函数调用子系统。为了使多个函数调用子系统按照设定顺序执行，先将一个 Function-Call Generator 模块与 Demux 模块相连接(Demux 模块的输出端口数量应与需要控制的函数调用子系统数量相等)；然后将 Demux 模块的输出端口与需要控制的系统相连。连接 Demux 第一个输出端口的系统先执行，连接第二个端口的次执行，以此类推。

2. 参数和对话框

Function-Call Generator 模块的参数对话框如图 7.140 所示。

- Sample time：两次采样的时间间隔。

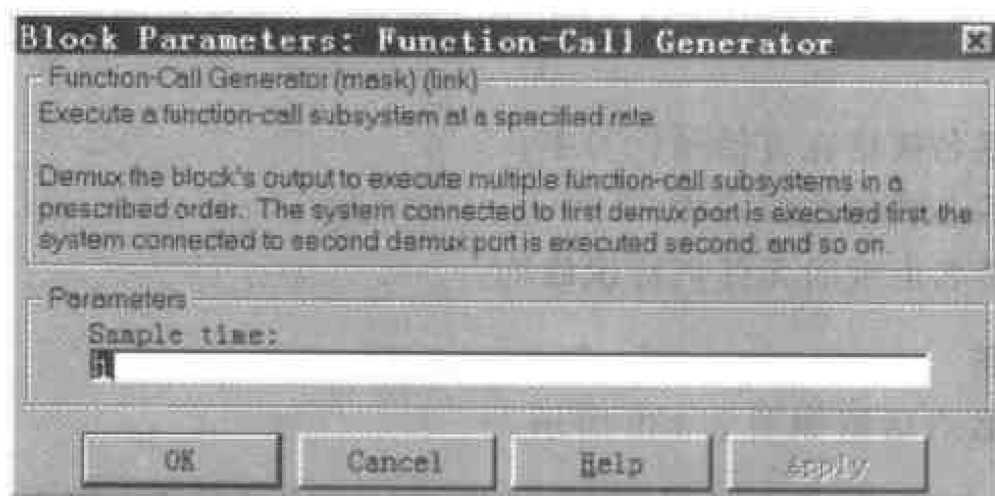


图 7.140　Function-Call Generator 模块的参数对话框

3. 特性

- 数据类型：输出双精度型实信号；
- 采样时间：用户指定；
- 可矢量化。

7.11.11　Goto 模块

1. 功能描述

Goto 模块将其输入传递给相应的 From 模块。输入可以是一个实或复信号，也可以是任意类型的矢量。参见“From 模块”。

一个 Goto 模块可以将其输入传递给多个 From 模块，但一个 From 模块只能接受一个来自 Goto 模块的信号。Goto 模块的输入被传递给相应的 From 模块的效果就好像这两个模块是连接在一起似的。至于 From 和 Goto 模块的使用限制，请参看“From 模块”。通过定义 Goto 模块标签的 Tag 参数使 Goto 模块和 From 模块配合使用。

Tag visibility 参数决定 From 模块在使用信号时是否受限：

- local(默认值)指使用相同标签的 From 和 Goto 模块必须在同一个子系统中。方括号([])里为局部标签名；
- scoped 指使用相同标签的 From 和 Goto 模块必须在同一个子系统中或在层次模型 Goto Tag Visibility 模块的下级子系统中。大括号({})内为区域标签名；
- global 指使用相同标签的 From 和 Goto 模块可以处在模型中的任何位置。

若使用相同标签名的 Goto 和 From 模块驻留在同一个子系统时，使用 local。若使用相同标签名的 Goto 和 From 模块驻留在不同的子系统时，请用 global 或 scoped 标签。若用户将一个标签定义为 global 后，所有使用该标签的模块均将得到同一个信号。定义为 scoped 标签可用在模型多处。一个使用两个具有相同名(A)标签的模型如图 7.141 所示。

2. 参数和对话框

Goto 模块的参数对话框如图 7.142 所示。

- Tag：Goto 模块的标签。该参数鉴别 Goto 模块定义范围。
- Tag visibility：Goto 模块标签的 local，scoped，global。缺省值为 local。

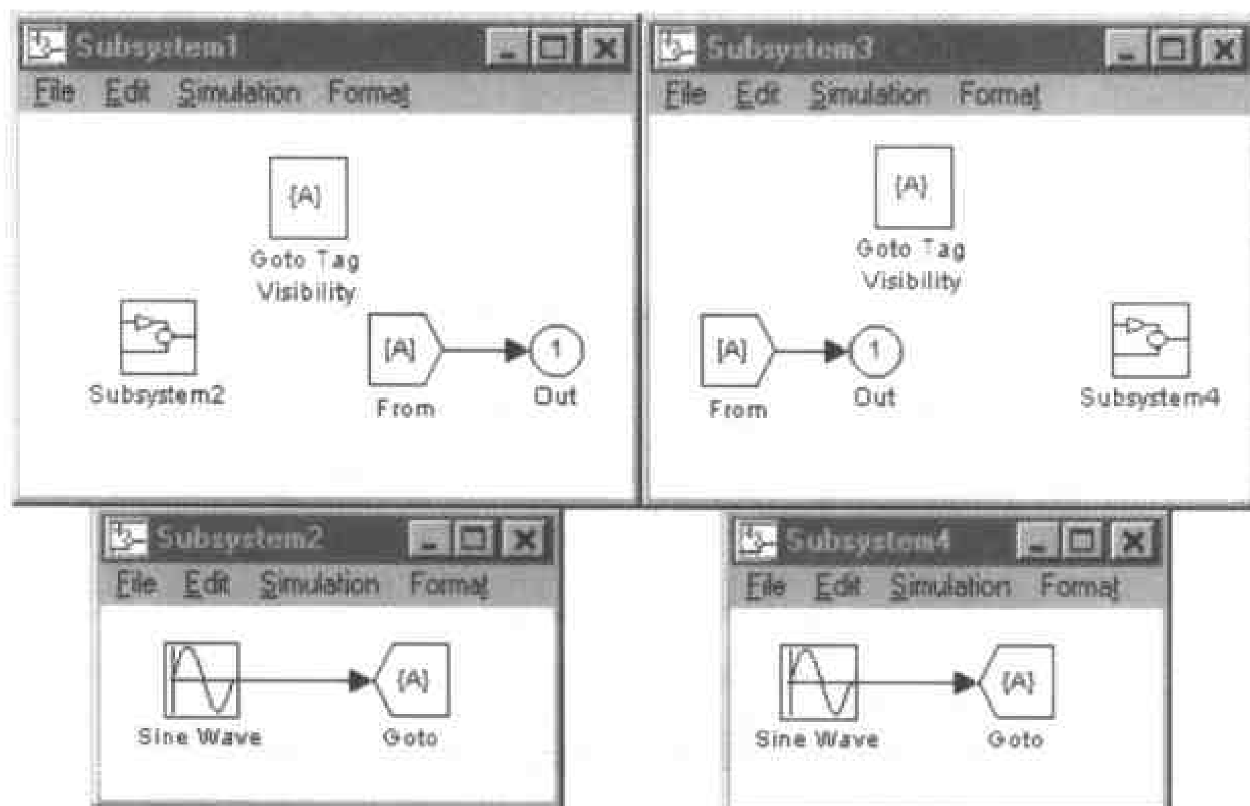


图 7.141　一个使用两个具有相同名(A)标签的模型

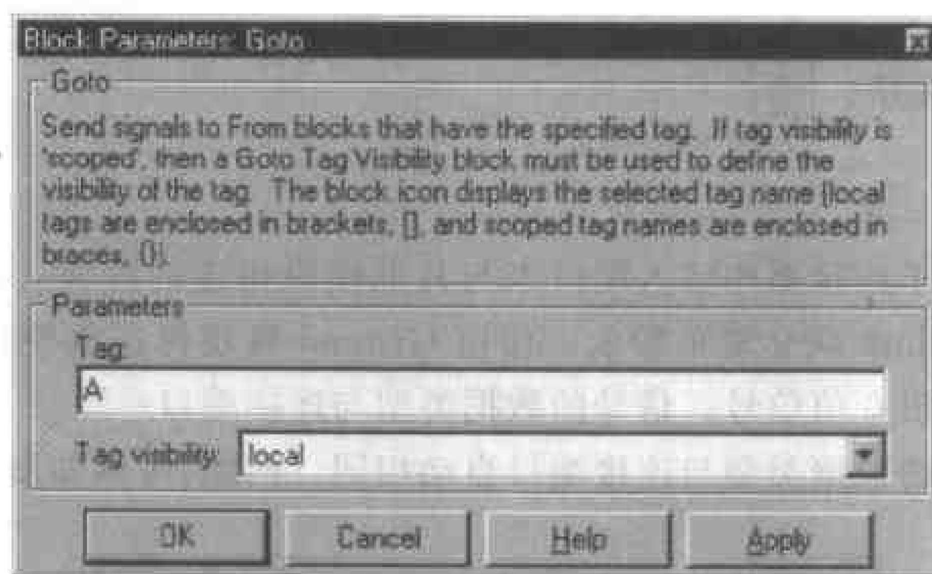


图 7.142　Goto 模块的参数对话框

3. 特性

- 数据类型：Goto 模块接受任意类型的实或复信号；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.11.12　Goto Tag Visibility 模块　{A}

1. 功能描述

Goto Tag Visibility 模块定义 scoped 的 Goto 模块标签可见的可达性。定义为 Goto tag 参数的标签，对处于包含 Goto Tag Visibility 模块的子系统或在层次模型的下一级子系统中的 From 模块来说，是可达的。对于 Tag Visibility 参数为 scoped 的 Goto 模块，要求有 Goto

Tag Visibility 模块。若标签参数是 local 或是 global，则该模块无用。模块的图标显示被括在{}内的标签名。

2. 参数和对话框

Goto Tag Visibility 模块的参数对话框如图 7.143 所示。

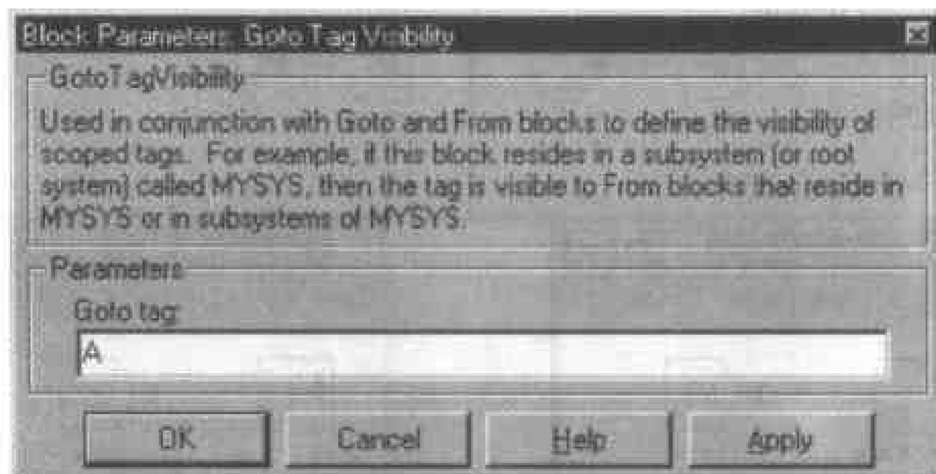


图 7.143　Goto Tag Visibility 模块的参数对话框

- Goto tag：Goto 模块标签，其可见度由本模块的位置定义。

7.11.13 Ground 模块

1. 功能描述

Ground 模块可用于连接那些输入端口未与其他模块相连的模块。若用户运行一个带有这样模块的模型，Simulink 就会发布警示。使用 Ground 模块将这些模块'接地'可避免警示出现。Ground 模块输出 0 值信号。信号的数据类型与连接端口一致。

Ground 模块输出的数据类型与连接端口自动相同。例如，考察如图 7.144 所示的模型。

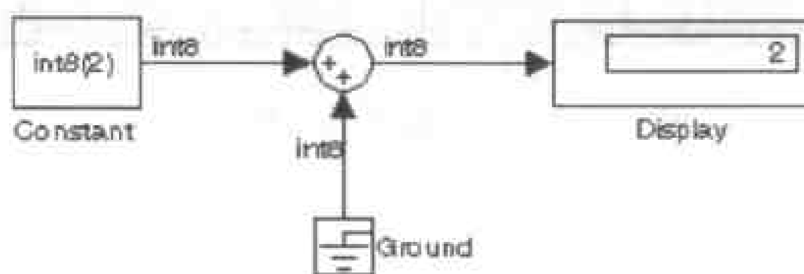


图 7.144　使用 Ground 模块举例

其中，Constant 模块输出的数据类型是 int8，这样 Ground 模块输出的数据类型也是 int8。

2. 参数和对话框

Ground 模块的参数对话框如图 7.145 所示。

3. 特性

- 采样时间：从驱动模块继承；
- 可矢量化。

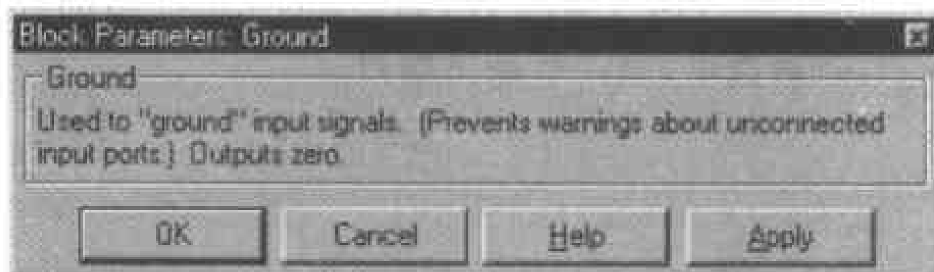
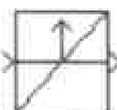

Block Parameters: Ground
Ground
Used to "ground" input signals. (Prevents warnings about unconnected input ports.) Outputs zero.
OK Cancel Help Apply

图 7.145 Ground 模块的参数对话框

7.11.14 Hit Crossing 模块 

1. 功能描述

Hit Crossing 模块探测在由 Hit crossing direction 参数定义的方向上输入到达 Hit crossing offset 参数值的时间。本模块在允许误差内找出转移到并跃过偏置的交会点。

模块接受一个双精度型输入信号。若 Show output port 复选框被选中，模块输出显示交会点出现的时间。如果输入信号恰好是偏置值，模块在该时间步内输出 1 值。如果偏置值在输入信号的两个相邻值之间(两者均不等于偏置值)，模块将在下一个时间步显示 1 值。若 Show output port 复选框未选中，模块确保仿真找到交会点但不产生输出。

Hit Crossing 模块在解决有穷数学和计算机精度等方面是非常有用的。对于需要在模型中加入逻辑判断来说，该模块可能更方便。

编者提示：读者可以从 Hardstop 和 clutch 演示程序了解 Hit Crossing 模块的用途。在 hardstop 中，Hit Crossing 模块位于 Friction Model 子系统中。在 Clutch 中，Hit Crossing 模块位于 Lockup Detection 子系统中。

一般情况下，Hit Crossing 模块输出双精度型信号。只有在激活了 boolean 兼容性时(参看第 3 章"激活严格布尔型检测")，模块才会输出布尔型信号。

2. 参数和对话框

Hit Crossing 模块的参数对话框如图 7.146 所示。

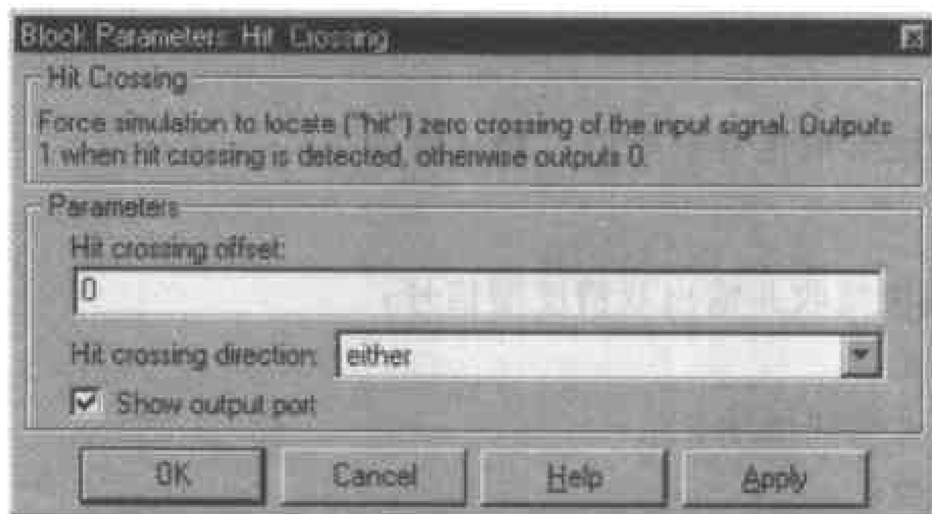

Block Parameters: Hit Crossing
Hit Crossing
Force simulation to locate ("hit") zero crossing of the input signal. Outputs 1 when hit crossing is detected, otherwise outputs 0.
Parameters
Hit crossing offset:
0
Hit crossing direction: either
Show output port
OK Cancel Help Apply

图 7.146 Hit Crossing 模块的参数对话框

- Hit crossing offset：需探测的交会点值。

- Hit crossing direction：输入信号逼近 Hit crossing offset 值的方向。
- Show output port：若被选中，画一个输出端口。

3. 特性

- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展；
- 可矢量化；
- 过零检测：探测交会点。

7.11.15 IC 模块

1. 功能描述

IC 模块对与模块输出端口相连的信号进行初始值设定。图 7.147 所示的这些模块描述了 IC 模块是如何初始化一个标签为 test signal 的信号的。在 $t=0$ 时，信号值是 3；然后，是 6。IC 模块在给环路中代数状态变量设置假设值等方面也是有用的。

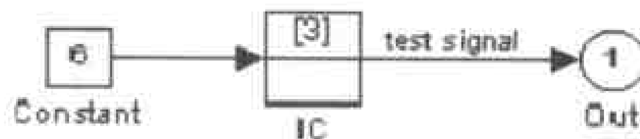


图 7.147 IC 模块应用举例

2. 参数和对话框

IC 模块的参数对话框如图 7.148 所示。

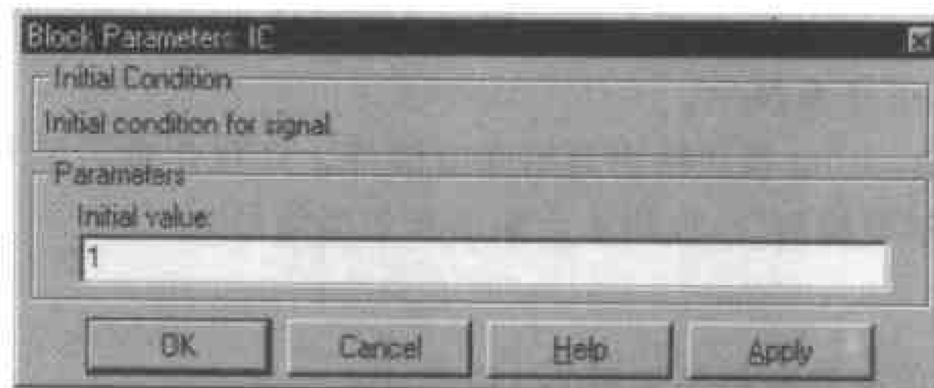


图 7.148 IC 模块的参数对话框

- Initial value：欲设置的信号初始值。默认值为 1。

3. 特性

- 数据类型：IC 模块接收并输出双精度型信号；
- 直通输出；
- 采样时间：从驱动模块继承；
- 标量扩展：仅参数；
- 状态数：0；
- 可矢量化。

7.11.16 Inport 模块 (1)

1. 功能描述

本版本标为 In1 模块。将系统与外界连接起来就是通过 Inport 模块与 Outport 模块一起实现的。Simulink 按照如下的规则对该模块的输入端口数赋值：

- 从 1 开始自动将在系统或嵌套子系统中的 Inport 模块进行顺序编号；
- 若用户添加一个 Inport 模块，该输入端口的编号为下一个有效编号；
- 若用户删除了一个 Inport 模块，以后的端口将递补编号。
- 若用户将一个 Inport 模块拷贝到一个系统中，除非其端口编号与在当前系统中的某个 Inport 模块的编号冲突，否则其端口将不重新编号。如果拷的 Inport 模块的端口编号不是顺序的，用户必须重新对模块进行编号，否则当用户运行模型仿真或修改结构图时就会产生错误信息。

如果 Inport 模块提供一个矢量信号，用户可以通过模块的 Port width 参数指定输入的宽度，或者让 Simulink 将其自动赋值为 −1(默认值)。Sample time 参数表示信号进入系统的速率。默认值 −1 表明模块将继承驱动模块的采样率。将 Inport 模块的参数设置为此参数，在最高级系统中或是在某个模型中 Inport 模块受控子采样时间不定的模块时是很合适的。

(1) 子系统中的 Inport 模块

子系统中的 Inport 模块相当于子系统的输入。Subsystem 模块的输入信号来自于该子系统关联的 Inport 模块。若 Inport 模块的 Port number 参数与 Subsystem 模块的相对位置匹配，则 Inport 模块就与 Subsystem 模块相关联。例如，Port number 参数为 1 的 Inport 模块从与 Subsystem 模块的最高级端口连接的模块获得信号。

如果用户改变 Inport 模块的 Port number 参数，尽管模块的输入信号仍然来自子系统外的同一模块，但模块已与不同的输入端口相连。

Inport 模块名以端口标签形式在 Subsystem 模块的图标上显示。如需取消显示标签，请选中 Inport 模块并选择 Format 菜单的 Hide Name。然后选择 Edit 菜单的 Update Diagram。

(2) 最高级系统中的 Inport 模块

最高级系统的 Inport 模块有两个用途：一是从工作空间获得外输入，用户可通过使用 Simulation Parameters 对话框或 sim 命令实现；二是提供一个模型摄动的分析手段，例如：

- 从工作空间获得外输入，使用 Simulation Parameters 对话框(参看"从基本工作空间载入输入")或 sim 命令的 ut 自变量(见"sim 命令")；
- 通过 linmod 和 trim 分析函数提供一种模型摄动的手段。Inport 模块定义注入系统的输入指向。欲获详情，请参看第 5 章中有关内容。

编者提示：在任何一个模型中，有一个 Inport 模块就必须有一个 Outport 模块。另外 Simulink4.X 版本已经将 Inport 模块和 Outport 模块配对放入各种 Subsystem 模块中。

2. 数据和数值类型

Inport 模块接受任意类型的实或复信号。一个 Inport 模块的输出数据和数值类型与相应的输入信号相同。用户必须通过模块的 Signal type 和 Data type 参数指定进入根级输入端口的外(指工作空间)输入的信号类型和数据类型。

连接根级输入端口的信号矢量的各元素，必须具有相同的数值和数据类型。如果子系统中包含了一个 Enable 或 Trigger 模块，且 Inport 模块直接与一个 Outport 模块相连，则输入信号各元素必须具有相同的数值和数据类型。除此之外，连接子系统输入的信号元素可以有不同的数值和数据类型。

在图 7.149 中，连接 In1 的矢量信号的元素具有相同的类型。而连接 In2 的矢量信号元素可以是不相同的。

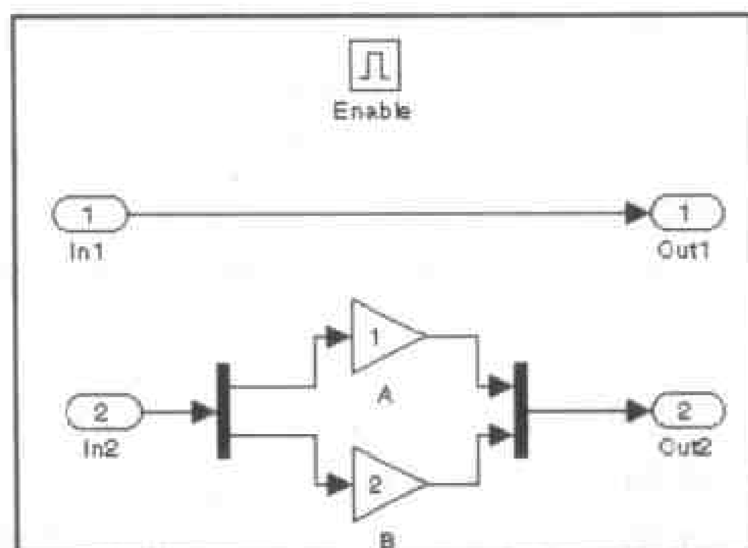


图 7.149　Inport 模块应用举例

3. 参数和对话框

Inport 模块的参数对话框如图 7.150 所示。

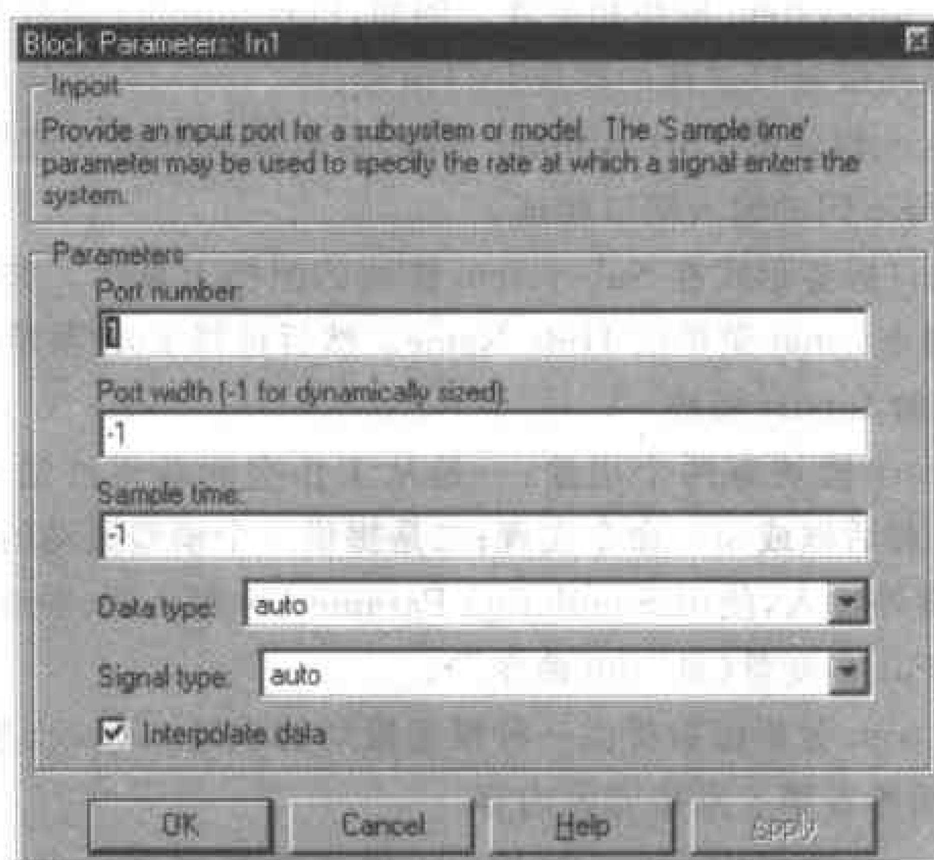


图 7.150　Inport 模块的参数对话框

- Port number：Inport 模块的端口数。
- Port width：注入 Inport 的信号宽度。指定－1 时系统自动决定。
- Sample time：进入系统时的信号速率。

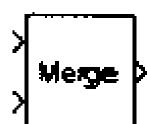
注意：下面三个参数仅适用于根级输入端口。不在子系统输入端口对话框中显示。

- Data type：外输入的数据类型。
- Signal type：外输入的信号类型（实或复）。
- Interpolate data：选择此项时，模块在没有相应工作空间数据存在的时间步内对输出进行内插或外插运算。

4. 特性

- 采样时间：从驱动模块继承；
- 可矢量化。

7.11.17 Merge 模块



1. 功能描述

Merge 模块将输入合成为一个行输出，在任意时刻，它等于驱动模块最新计算出的输出。用户可以通过模块的 Number of Inputs 参数定义任意输入数。所有输入的宽度必须相同。例如，所有标量和矢量的宽度均为 3。用户可以通过 Initial Output 参数指定一个初始输出。若用户不指定，而一个或多个驱动模块指定了初始输出，则 Merge 模块的初始输出将等于驱动模块最新求得的输出。Merge 模块使创建交替执行的子系统变得更容易。参看第 6 章"创建交替执行子系统"的内容。

Simulink 严格限制用户能用于 Merge 模块输入的连接种类。具体地讲，Simulink 只允许建立一个从非虚拟模块的输出到 Merge 模块的输入之间的一对一映射的连接。例如，用户可以用 Go To/From 模块，将某一个模块图中的非虚拟模块的标量或矢量输出与另一个模块图中的 Merge 模块连接起来。图 7.151 举例说明了连接多个非虚拟模块和一个 Merge 模块的正确方法。

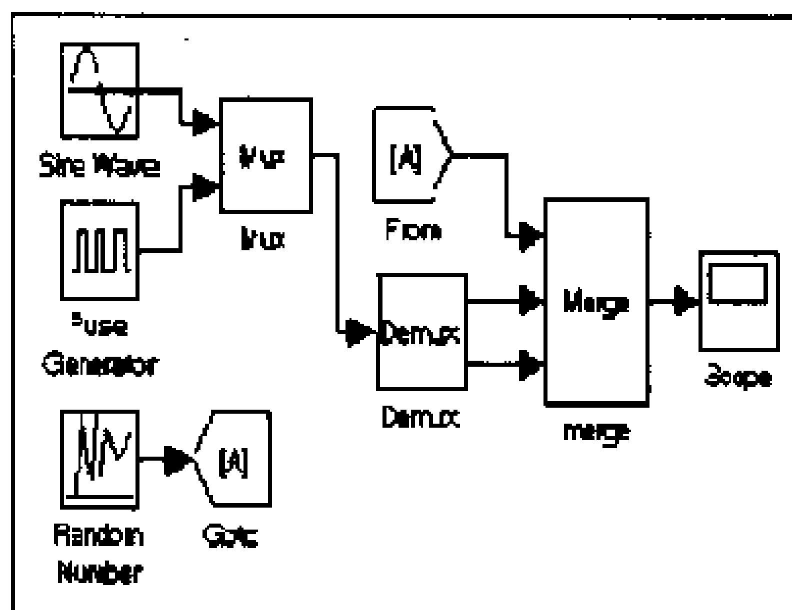


图 7.151 Merge 模块应用举例（正确接法）

用户可用一个 Merge 模块将多个非虚拟输出与一个 Merge 模块上的单个输入相连接。图 7.152 说明了多个非虚拟模块与一个 Merge 模块连接是无效的。

Simulink 在仿真开始前检测模块模型图的无效连接，每探测到一个就停止下来显示一个

错误信息。

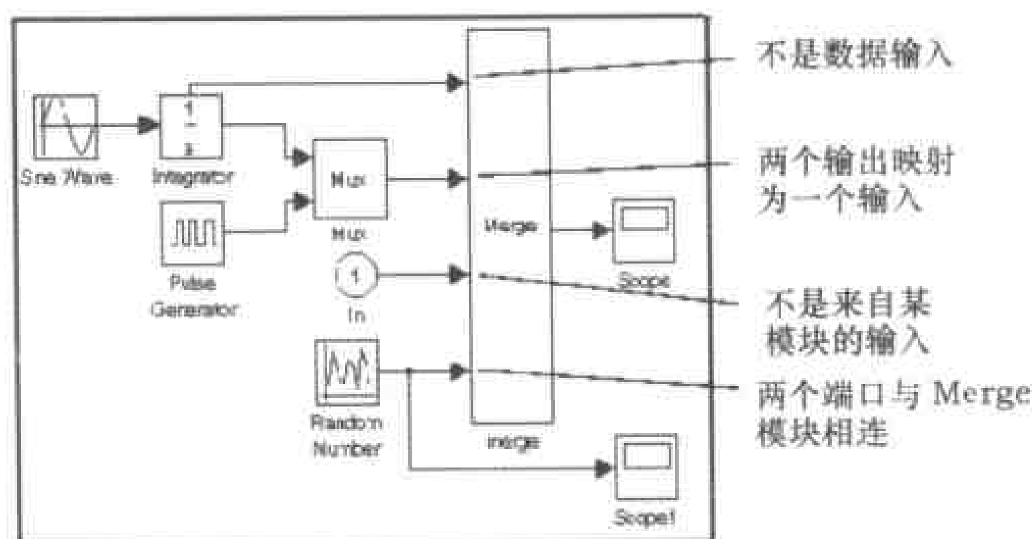


图 7.152　Merge 模块应用举例(无效接法)

2. 参数和对话框

Merge 模块的参数对话框如图 7.153 所示。

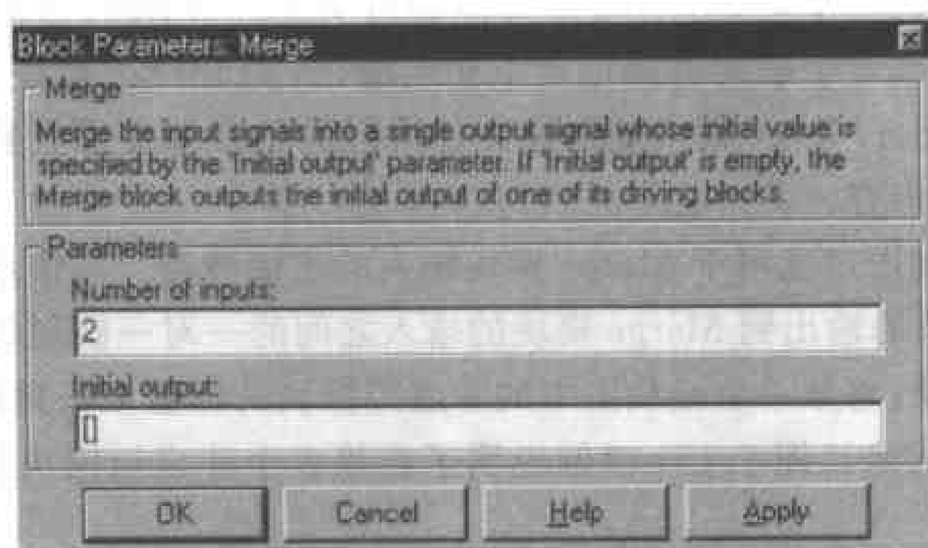


图 7.153　Merge 模块的参数对话框

- Number of inputs：欲合并的输入端口数。端口可以是标量或矢量。
- Initial output：输出的初始值。若未指定，初始输出就等于驱动模块的初始输出。

3. 特性

- 数据类型：Merge 模块接受包括用户定义的任意数值型(复或实)和数据类型的信号。若输入是由用户定义的，初始条件必须为 0；
- 采样时间：从驱动模块继承；
- 可矢量化。

7.11.18　Model Info 模块

1. 功能描述

Model Info 模块以在模型图中注释模块的方式显示模型的版本控制信息。图 7.154 举例

说明了 Model Info 模块关于 vdp 模型信息显示的使用方法。

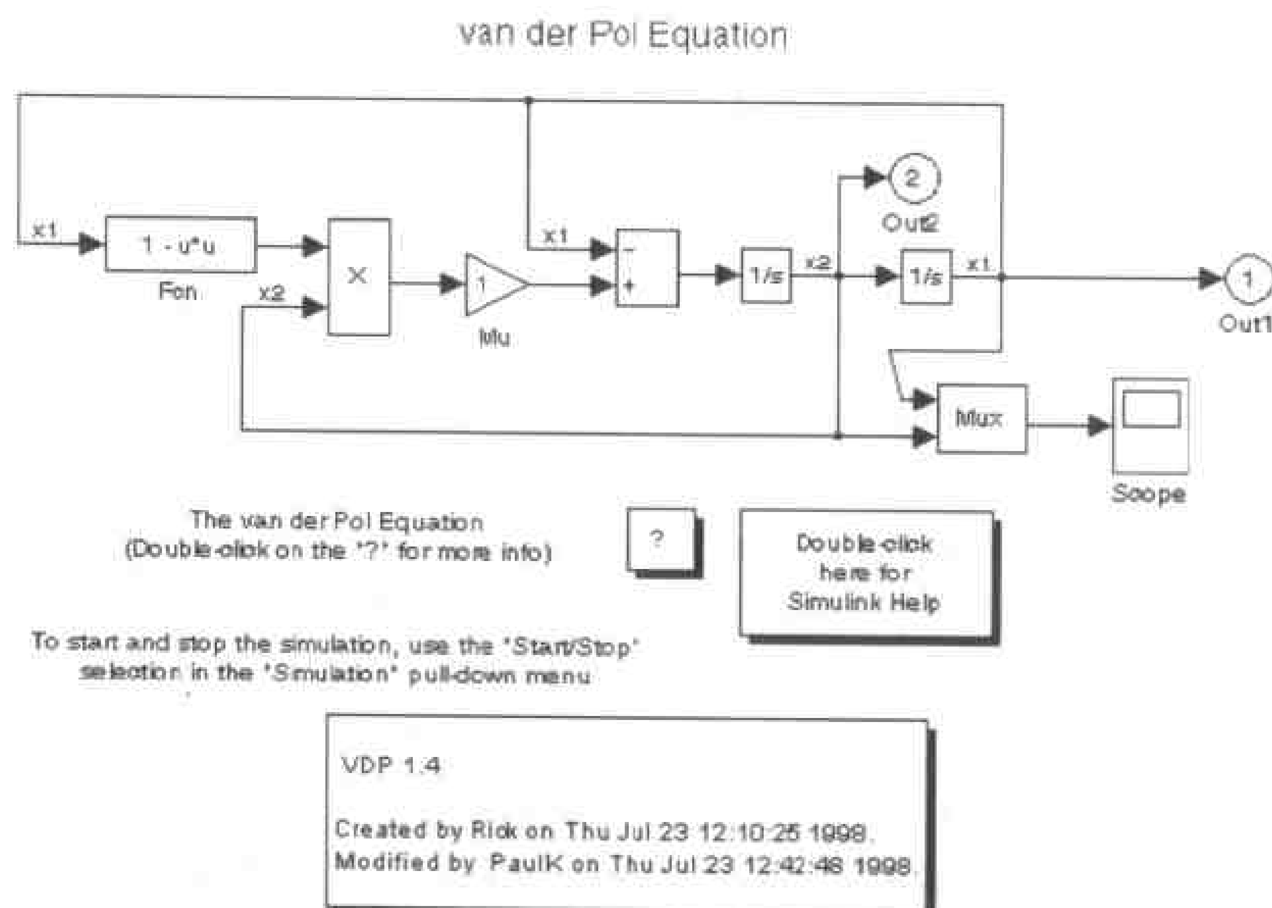


图 7.154 Model Info 模块使用举例

一个 Model Info 模块可以显示模型本身的版本控件信息和(或)由一个外部版本控件或构造操作系统所提供的信息。Model Info 模块的对话框允许用户指定模块显示文本的内容和格式。

2. 参数和对话框

Model Info 模块的参数对话框如图 7.155 所示。

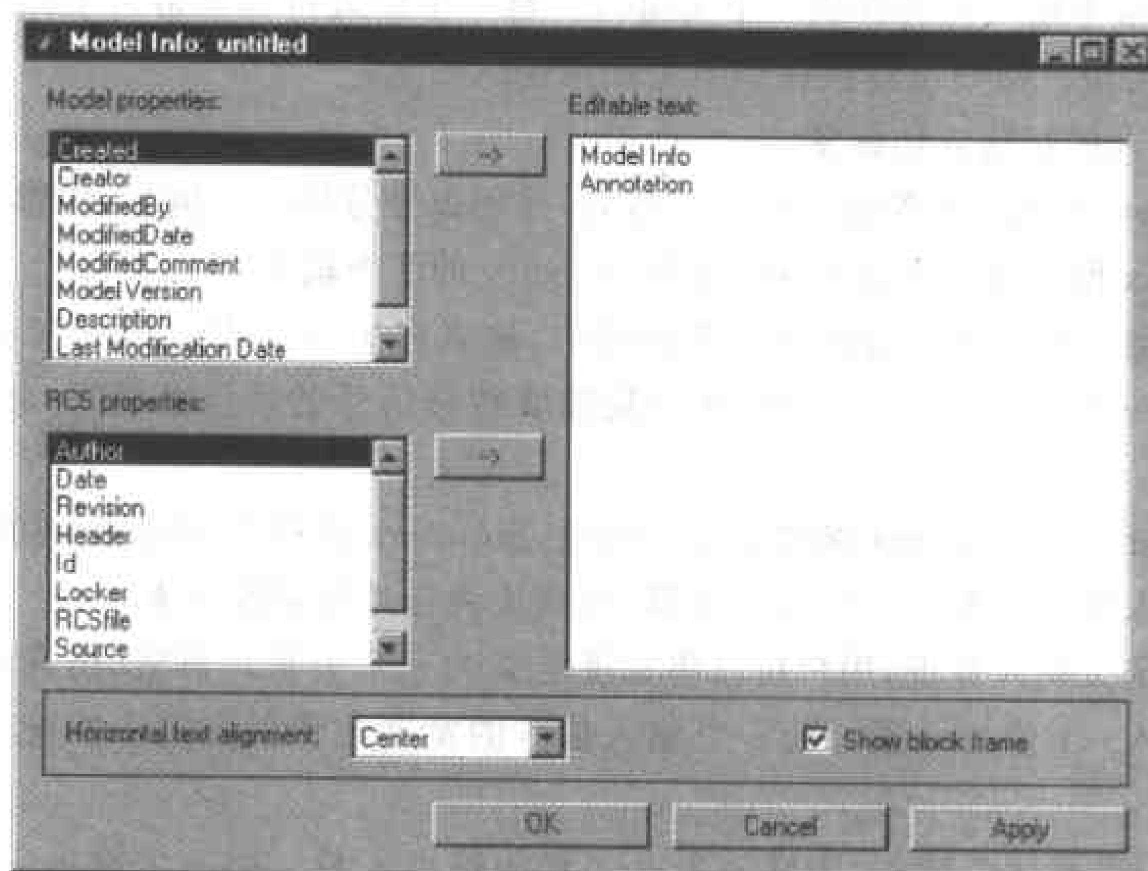


图 7.155 Model Info 模块的参数对话框

Model Info 模块的对话框包括下列各栏：

• Editable text 在此栏中输入 Model Info 模块显示的文本。用户可自由地嵌入形式为%<propname>的变量，其中，输入文本中的 propname 是一个模型名或版本控制系统属性。在显示文本中，属性值取代变量。例如，假设模型的当前版本是 1.1，然后输入文本

Version %<ModelVersion>

在显示文本中显示为：

Version 1.1

用户以这种方式能够涉及的模型和版本控制系统属性在 Model properties 栏和 Configuration manager properties 栏中列出。

• Model properties 列出存在模型中的版本控制属性。选择一个属性，然后选中相邻的箭头按钮，在 Editable text 栏中输入相应的变量。例如，选中 CreatedBy，在 Editable 文本栏中输入%<CreatedBy%>。欲知在本栏指定属性的使用说明，请参看第 3 章"版本控制属性"。

• RCS properties 本栏只有当用户事先为本模型定义了一个外部设置管理器才会出现。其标题随所选的设置管理器而变（如 RCS Properties）。本栏列出由用户能够放在 Model Info 模块中的外系统所支持的版本控制信息。欲从表中放入某项目，请选中，然后点击相邻的箭头按钮。

注意：只有用户将模型编辑完，关闭再重新打开模型后，所选的项目才会出现。

7.11.19 Mux 模块

1. 功能描述

Mux 模块将多个输入行合并成一个矢量行。每一个输入行可携带一个标量或矢量信号。模块输出为一个矢量。用户可以根据如下方法给输入命名：

• 给代表引入的行输入加标签。

• 在 Number of inputs 参数内输入信号名，并用逗号分隔开。例如，在参数中输入 position，velocity，Mux 模块就有名为 position 和 velocity 的两个输入。

对于一个未加标签的输入行或未连接的端口，缺省名为 signalN，其中 N 是输入端口数。当用户正在定义一个可使用 Bus Selector 模块抽取特殊信号的信号总线时，此选项是很有用的。

若用户将 Number of inputs 参数定义为标量，Simulink 将核对 Mux 模块的输出端口以决定送入模块的输入宽度。若一个输入为矢量，则模块将合并矢量各元素。

若需要确切定义输入宽度，用户可以将宽度定义为一个矢量。再将（仿真过程中）自动确定宽度的输入放入−1 值的元素。若一个输入信号的宽度与预期宽度不匹配，Simulink 将显示错误信息。

例如，[4 1 2]表示三个输入，构成一个七元素的输出矢量。前四个输出矢量元素来自第一个输入，第五个元素来自第二个输入，第六和第七个来自第三个输入。若固定这些输入宽度无关紧要，用户可将 Number of inputs 参数设置为 3。

为指定三个输入并且第一个输入矢量必须由四个元素组成，用户应指定[4 −1 −1]。Simulink 确定第二个和第三个输入的宽度并相应调整输出宽度的大小。

Simulink 画出的 Mux 模块包含了指定数量的输入端口。若用户改变了输入端口的数量，Simulink 将相应从模块图标的底部增加或删除。

• 使用一个变量以规定 Number of Inputs 参数。

当用户将 Number of inputs 参数指定为一个变量后，若此变量未在工作空间定义过，Simulink 将给出错误信息。

编者提示：当用户将 Mux 模块从 Simulink 模块库复制到一个模型中时，Simulink 不显示模块名。

2. 数据类型

Mux 模块接受包括混合型矢量在内的任意数据类型的实或复信号。

3. 参数和对话框

Mux 模块的参数对话框如图 7.156 所示。

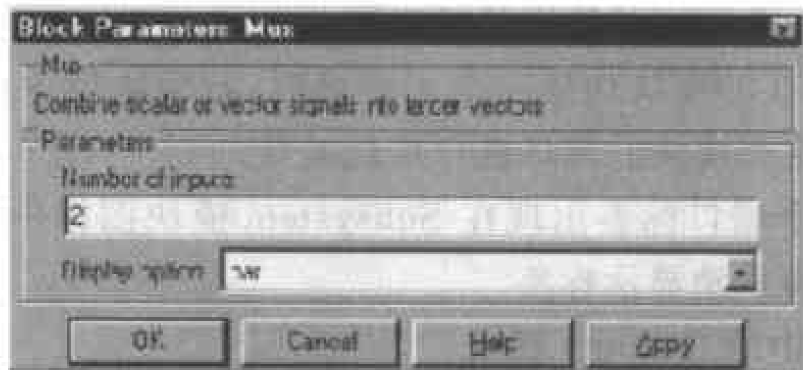


图 7.156 Mux 模块的参数对话框

• Number of inputs：输入的数量和宽度。行输出的宽度等于行输入宽度之总和。当 Display option 参数为 names 时，用户可在 number of inputs 参数栏内输入用逗号分隔的信号名表。

• Display option：用户模型中模块图标的外观。

表 7.16 图标外观与选项对应

Display Option	模型中模块图标的外观
None	Mux 显示在模块图标内
Names	在每一个端口显示信号名
bar	以实心前景色显示模块图标

7.11.20 Outport 模块

1. 功能描述

本版本标为 Out 1。Outport 是一个系统与系统外目标的链接。Simulink 根据如下规则对 Outport 模块端口数赋值：

- 自动对最高级系统或子系统内的 Outport 模块按顺序从 1 开始进行编号。
- 若用户添加一个 Outport 模块,赋予模块下一个有效编号。
- 若用户删除一个 Outport 模块,其他端口将自动依次重新编号以保证模块编号的顺序性和连续性。
- 若用户将一个 Outport 模块复制到一个系统中,除非其当前端口编号与已在该系统中的 Outport 模块编号相冲突,模块端口将不进行重新编号。若复制的 Outport 模块端口编号不是顺序的,用户必须对其进行重新编号,否则当仿真运行或更新模块框图时就会出现错误信息。

(1) 子系统中的 Outport 模块

子系统中的 Outport 模块起子系统的输出作用。进入一个子系统内的 Outport 模块的输入信号从相关 Subsystem 模块的输出端口输出。与 Subsystem 模块输出端口关联的 Outport 模块,其 Port number 参数与 Subsystem 模块输出端口的相对位置匹配。例如,Port number 为 1 的 Outport 模块将其信号送至与 Subsystem 模块最上方一个输出端口连接的模块。

若用户对 Outport 模块的 Port number 重新编号,虽然模块继续将信号送至子系统外的同一个模块,但模块将与一个不同的输出端口连接。

当用户通过选择已存在的模块构建一个子系统时,若模块组中包含了多于一个 Outport 模块,Simulink 将自动对这些模块的端口进行重新编号。

Outport 模块名以一个端口标签出现在 Subsystem 模块图标中。选中模块并从 Format 菜单上选择 Hide Name,可取消显示标签。

(2) 条件执行子系统中的 Outport 模块

当一个 Outport 模块处于一个触发和(或)使能子系统中时,用户可以在子系统处于禁止状态下指定输出:恢复(reset)初始值或保持(held)在当前值。Output when disabled 下拉菜单提供这些选项。Initial output 参数是在子系统执行之前的输出值(子系统处于禁止状态并且 there set 选项被选中)。

(3) 最高级系统中的 Outport 模块

最高级系统中的 Outport 模块有两个用途:一个给工作空间提供外部输出,用户可通过使用 Simulation Parameters 对话框或 sim 命令实现;二是为分析函数获得系统输出提供一种手段。

- 给工作空间提供外部输出,使用 Simulation Parameters 对话框(参看第 4 章"保存进入工作空间的输出")或 sim 命令(参看第 4 章"sim 命令")。例如,若一个系统包含多于一个 Outport 模块,如下的命令:

```
[t,x,y] = sim(...);
```

把 y 写为一个矩阵,矩阵每一列包含了一个不同 Outport 模块的数据。列的顺序与 Outport 模块的端口数的顺序一一对应。

若在第二个状态自变量之后用户指定了多于一个变量名,每一个 Outport 模块的数据将写给一个不同的变量。例如,若系统有两个 Outport 模块,将 Outport 模块 1 的数据存入 speed,Outport 模块 2 的数据存入 dist,用户可用下列命令:

```
[t,x,speed,dist] = sim(...);
```

- 为 linmod 和 trim 分析函数获得系统输出提供一种手段。欲获得更多有关使用带有分

析命令的 Outport 模块的信息，请参看第5章。

2. 数字和数据类型

一个 Outport 模块接受任意 MATLAB 数据型输入的实或复信号。模块输出的数字和数据类型与输入相同。一个与 Outport 模块连接的信号矢量的元素可以具有不同的数字和数据类型，但下列情况除外：若输出端口处于一个条件执行子系统中，并且初始输出没有被指定，则输入矢量的所有元素必须具有相同的数字和数据类型。

Simulink 的数据转换规则适用于 Outport 的 Initial output 参数。若初始值在模块输出数据类型的范围内，Simulink 将把初始值转换为输出数据类型。若转换使精度降低，Simulink 将提示警告信息。若指定的初始值超出了输出数据类型，Simulink 将暂停仿真并给出错误信息。

注意：模块的输出数据类型就是连接模块输入的信号数据类型。

3. 参数和对话框

Outport 模块的参数对话框如图 7.157 所示。

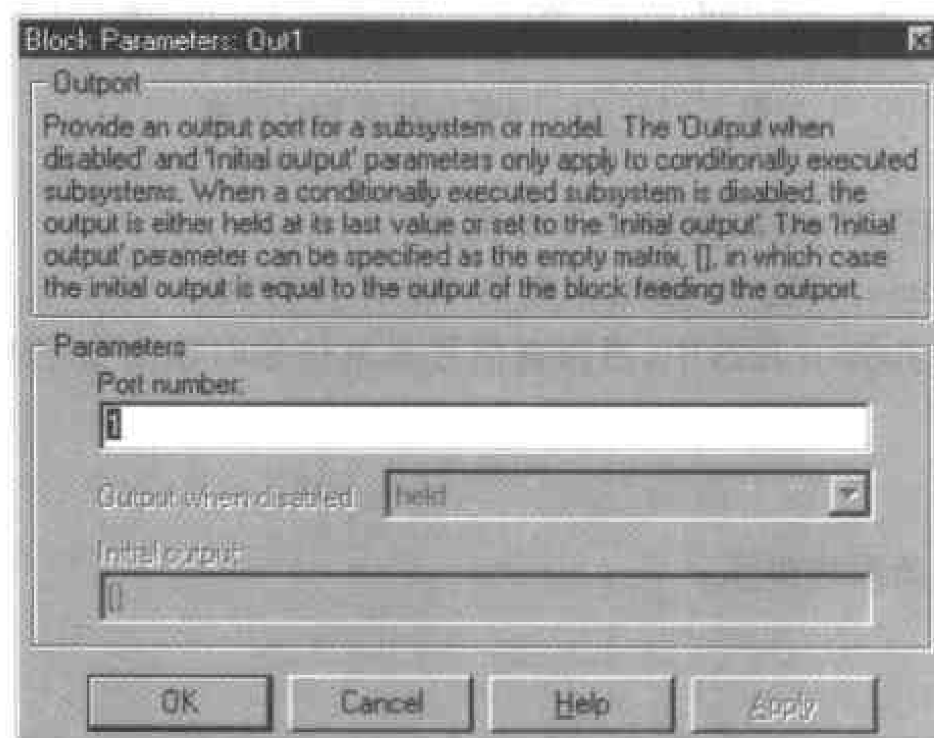


图 7.157 Outport 模块的参数对话框

- Port number：Outport 模块的端口数。
- Output when disabled：对于条件执行子系统，当系统未被激活时模块输出的发生事件。
- Initial output：对于条件执行子系统，在子系统执行前和未被激活时模块的输出。

4. 特性

- 采样时间：从驱动模块继承；
- 可矢量化。

7.11.21 Probe 模块



1. 功能描述

Probe 模块输出关于其输入信号的有关信息。模块可将输入信号的宽度、采样时间和输

入是否为复信号的标志等输出。模块只有一个输入端口。输出端口数取决于用户选择探查信息(即信号宽度、采样时间和复信号标志等)的数量。每一个探查值以独立信号从相互独立的端口输出。模块接受任意定制数据类型的实或复信号或矢量。模块输出双精度型信号。在仿真过程中,模块图标显示探查数据。

2. 参数和对话框

Probe 模块的参数对话框如图 7.158 所示。

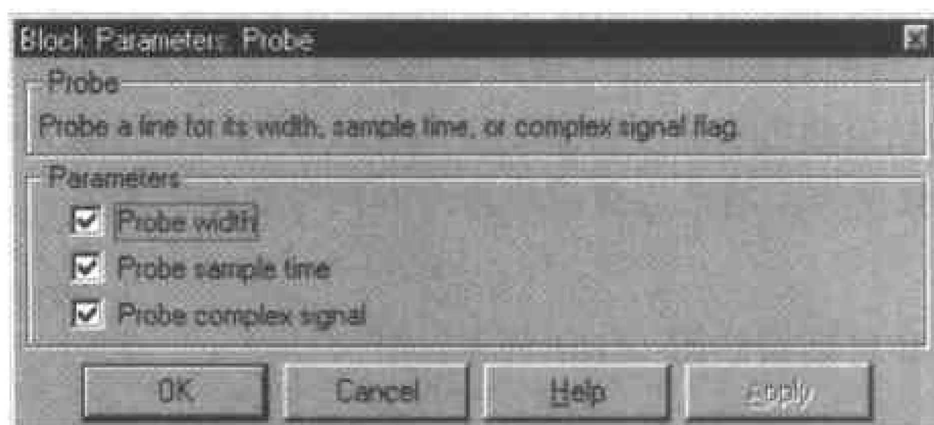


图 7.158　Probe 模块的参数对话框

- Probe width:若选中,输出探查信号的宽度。
- Probe sample time:若选中,输出探查信号的采样时间。
- Probe complex signal:若选中,当探查信号是复信号时输出 1;否则为 0。

3. 特性

- 直通输出;
- 采样时间:从驱动模块继承;
- 标量扩展;
- 可矢量化。

7.11.22　Selector 模块

1. 功能描述

Selector 模块对输入矢量的元素进行有选择的输出。

Elements 参数对输入矢量各元素的次序进行重新定义并作为输出矢量的次序。本参数必须指定为一个矢量,除非仅仅选择了一个元素。例如,如图 7.159 所示的是显示当输入矢量为[2 4 6 8 10]时的 Selector 模块图标和输出以及 Elements 参数为[5 1 3]的模型。

模块图标显示了输入矢量元素的次序。若模块不是很大,就显示模块名。

Selector 模块接受包括混合型在内的任意类型的信号。输出矢量元素与输入矢量所选择的相应元素的数据类型相同。

2. 参数和对话框

Selector 模块的参数对话框如图 7.160 所示。

- Elements:出现在输出矢量中的输入矢量元素的次序。

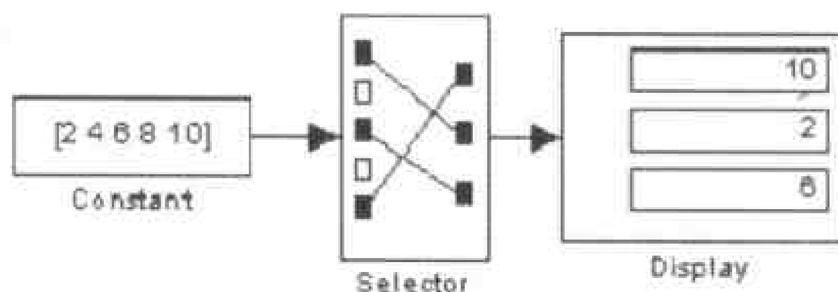


图 7.159　Selector 模块应用举例

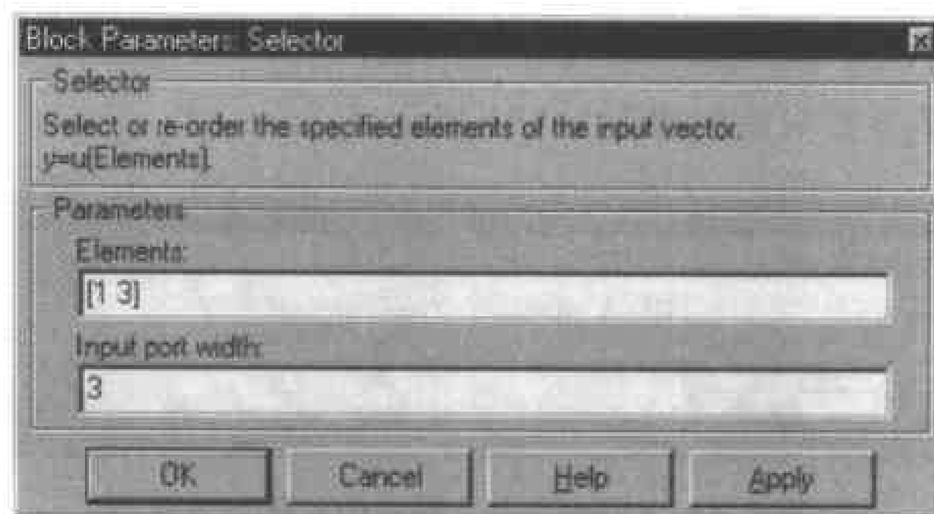


图 7.160　Selector 模块的参数对话框

- Input port width：输入矢量中元素的数量。

3. 特性

- 采样时间：从驱动模块继承；
- 可矢量化。

7.11.23　Subsystem 模块 

1. 功能描述

本模块相当于一个系统中的系统。用户可以采用如下方法构建一个子系统：

- 将连接库中的 Subsystem 模块拷贝到用户的模型中，然后打开模块并将其复制到子系统窗口，将模块加入到子系统中。
- 用约束框选择构建子系统的模块和连线，然后选中 Edit 菜单中的 Create Subsystem 选项。当用户打开模块时，窗口就会显示用户所选择的模块，并添加了反映进出子系统信号的 Inport 和 Outport 模块。

在 Subsystem 模块图标上画出的输入端口数与子系统中 Inport 模块数相对应。类似地，画在模块上的输出端口数与子系统中 Outport 模块数相对应。若 Inport 和 Outport 模块名不是隐含的，它们将以端口标签的形式出现在 Subsystem 模块上。请参看第 3 章“构建子系统”。

2. 特性

- 数据类型：参看“Inport 模块”和“Outport 模块”；
- 采样时间：取决于子系统中的模块；
- 矢量化：取决于子系统中的模块；
- 过零检测：若存在“使能”和“触发”端口。

7.11.24 Terminator 模块

1. 功能描述

Terminator 模块用于匹配输出端口未与其他模块连接的模块。若用户运行含有未连接输出端口模块的仿真时，Simulink 将给出警告。使用 Terminator 模块作为这些模块的终端，可以避免这种情况。

2. 参数和对话框

Terminator 模块的参数对话框如图 7.161 所示。



图 7.161 Terminator 模块的参数对话框

3. 特性

- 采样时间：从驱动模块继承；
- 可矢量化。

7.11.25 Trigger 模块

1. 功能描述

给一个子系统加一个 Trigger 模块可以使子系统成为触发子系统。一个触发子系统，在一个集成步内，当通过触发端口的信号值以指定的方式发生变化时，执行一次。一个子系统只能含有一个 Trigger 模块。关于创建触发子系统，请参看第 6 章。

Trigger type 参数允许用户选择触发子系统执行的事件形式：

- rising，当控制信号从负或零值上升至某一个正值(或零值，若初始值为负)时触发子系统执行。
- falling，当控制信号从正或零值下降至某一个负值(或零值，若初始值为正)时触发子系统执行。
- either，当信号上升或下降时触发子系统执行。
- function-call，使子系统的执行由一个 S-function 函数内的逻辑控制。参看第 6 章"函数调用子系统"。

用户可以通过 Show output port 复选框输出触发信号。该选项允许系统决定引起触发的条件。信号宽度就是触发信号的宽度。信号值为：

- 上升沿触发信号，1。
- 下降沿触发信号，-1。
- 其他，0。

2. 参数和对话框

Trigger Port 模块的参数对话框如图 7.162 所示。

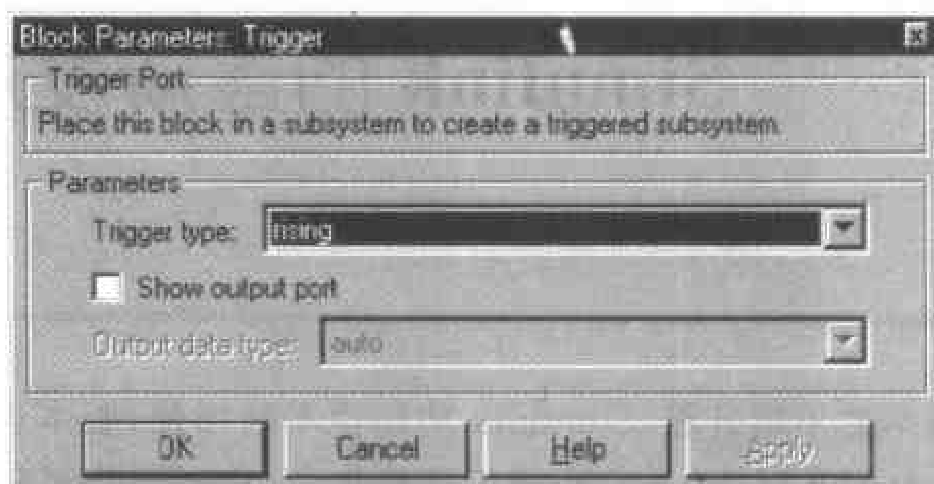
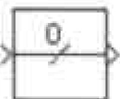


图 7.162　Trigger Port 模块的参数对话框

- Trigger type:触发子系统执行的事件形式。
- Show output port:若选中,Simulink 画出 Trigger 模块的输出端口并输出触发信号。
- Output data type:指定触发输出的数据类型(double 或 int8)。若用户选择 auto,Simulink 将把数据类型设置为与输出连接的端口的数据类型。若端口的数据类型不是 double 或 int8,Simulink 将给出错误信息。

3. 特性

- 数据类型:Trigger 模块接受布尔或双精度型信号;
- 采样时间:由触发端口的信号决定;
- 可矢量化。

7.11.26 Width 模块 

1. 功能描述

Width 模块把输入矢量的宽度作为输出。

Width 模块接受包括混合型信号矢量在内的任意数据类型的实或复信号。

2. 参数和对话框

Width 模块的参数对话框如图 7.163 所示。

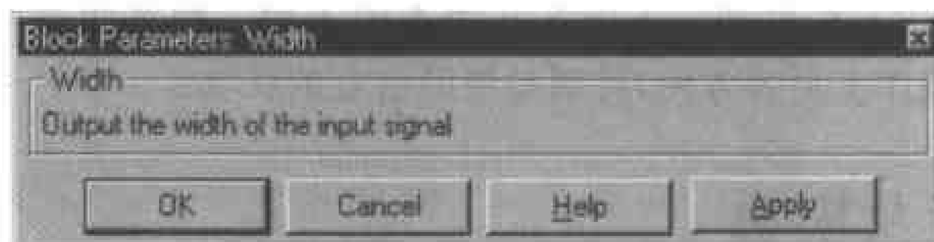


图 7.163　Width 模块的参数对话框

3. 特性

- 采样时间:常数;
- 可矢量化。

第8章 Simulink 的最新发展

随着 MATLAB 版本的不断升级，Simulink 功能也不断增强，版本也升级为 4. X。Simulink 4.0 是目前已推出的 MATLAB 6.0 中的工具箱之一，相对于前面介绍的基于 MATLAB 5.3 的 Simulink 3.0 而言，在功能上有了进一步的扩展。由于 Simulink 4.0 的出现，使得模型的开发和优化更加便利。我们编写本章的目的是希望通过对新增功能与性能改进的介绍，使读者了解 Simulink 的最新发展，并为需要了解新版本的读者提供参考。需要指出的是，本章所涉及的内容只适用于 Simulink 4. X，若读者使用的是 Simulink 3.0(基于 MATLAB 5.3)，则可忽略本章的内容。

编者提示：若读者涉及的模型不是十分复杂的话，最好使用运行速度较快的 Simulink 3.0 版本。同其他软件一样，在 Simulink 3.0 上建立的模型也可在 Simulink 4. X 版本上运行和保存，而且最令人高兴的是，Simulink 3.0 的用户不需要再为版本的运行环境苦恼，因为 Simulink 4.1(及以后版本)可将 Simulink 4.0 及以后版本模型以 Simulink 3.0 的格式保存。这也为 Simulink 3.0 版本的用户提供了极大的便利条件。另外，根据编者实践显示，Simulink 4.1 的稳定性比 Simulink 4.0 好。

8.1 进入 Simulink 4. X

启动 MATLAB，即可进入如图 8.1 所示的 Matlab 6.0 主窗口。

在命令窗口键入：

```
>>Simulink
```

或点击 Simulink 图标都可以进入 Simulink 库浏览器窗口(图 8.2)。其他操作基本同 3.0 版。

可以看出，Simulink 4.0 与 Simulink 3.0 版本的不同。相对而言，Simulink 4.0 版本较 Simulink 3.0 版本在操作上更为方便，特别是帮助信息的获取。

8.2 新增功能及模块

本节所介绍的内容是在 Simulink 3.0 基础上的新增或改进的功能。包括以下几个方面：

- Simulink 编辑器；
- 建模改进；
- Simulink 调试器；

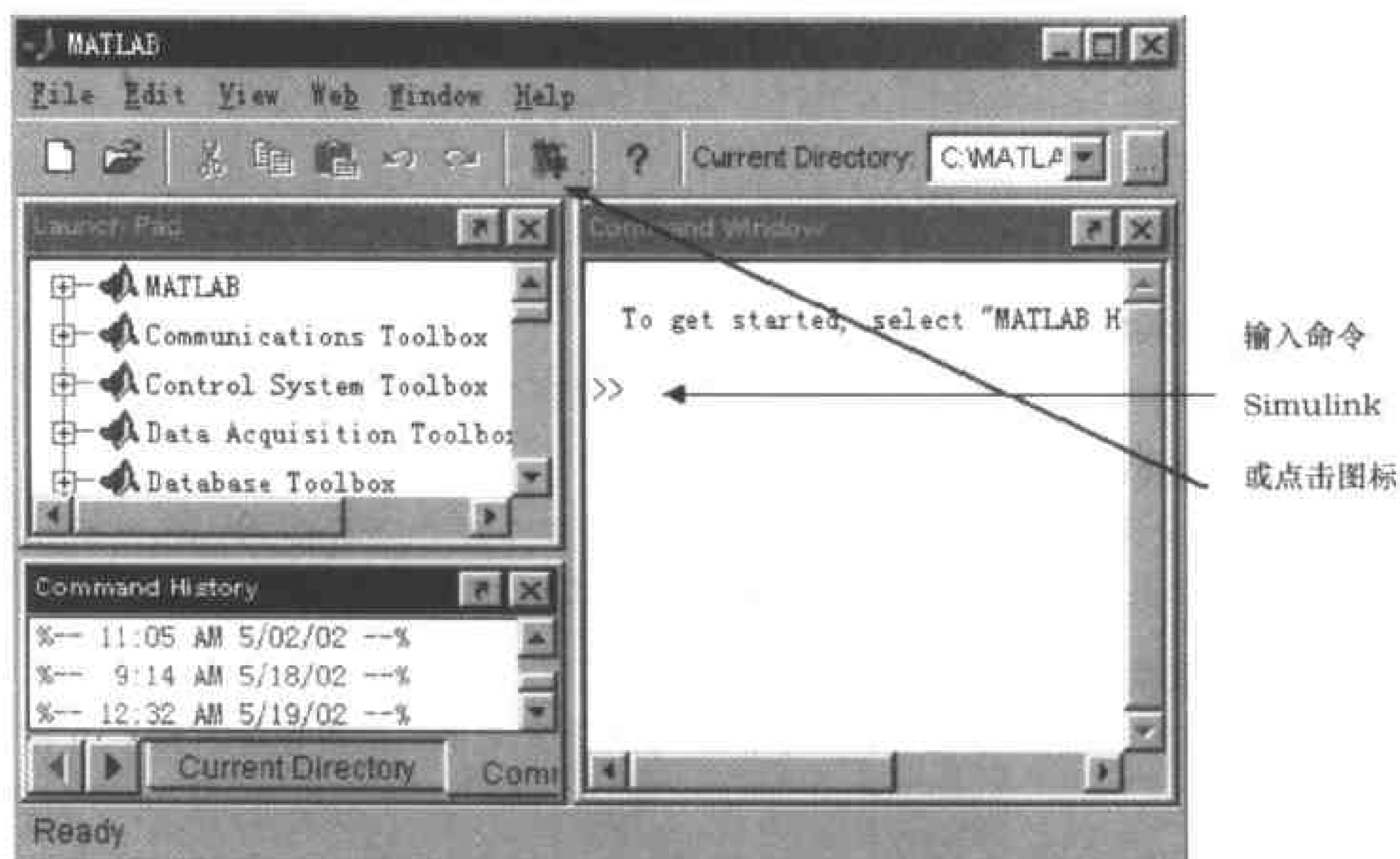


图 8.1　MATLAB 6.0 主窗口

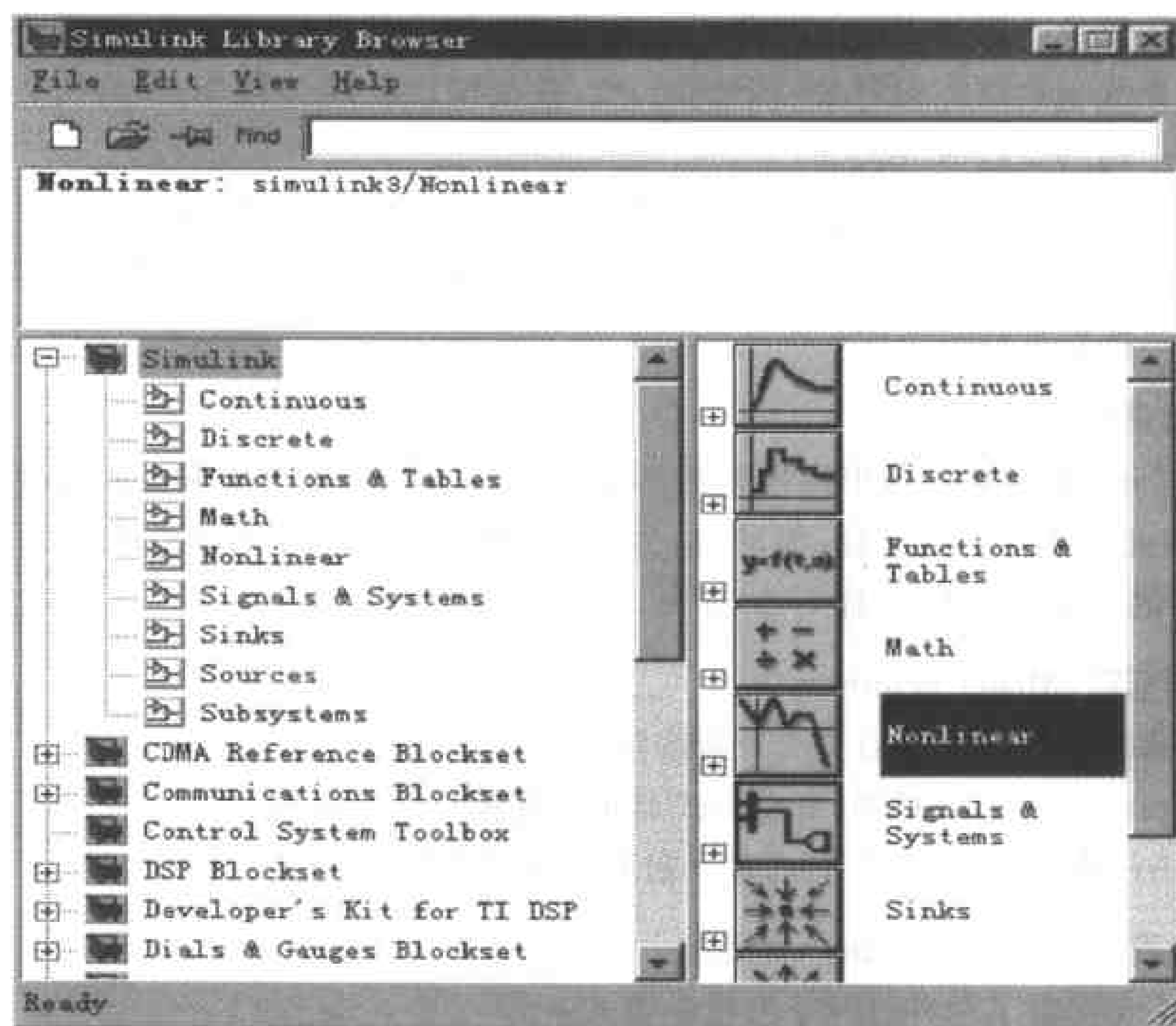


图 8.2　Simulink 4.0 库浏览器

- 模块库；
- SB2SL。

8.2.1 Simulink 编辑器

新版本的 Simulink 编辑功能在原有的基础上做了一些改进，体现在以下几个方面：

1. 参数选择(Preferences)

Simulink 4.0 的 Preferences 对话框允许用户指定多个选项的默认设置。

2. 文本对齐方式(Text Alignment)

Simulink 4.0 允许用户选择不同的注释文本的对齐方式。只要选中注释，然后从编辑器的菜单或上下文(右击)菜单上选中 Text Alignment 即可。

3. Unix 上下文菜单(Unix Context Menus)

Simulink 4.0 的 Unix 版本现在提供模块图的上下文菜单。点击鼠标右键即可显示该菜单。

4. 库链接(Library Link)

Simulink 4.0 在每一个表示模型库链接的模块上显示一个箭头符号。Simulink 4.0 也允许用户更改一个模型的链接，并将结果传回库。若用户要改变子系统的结构，就必须禁止标准模块同库模块的链接。假如用户试图修改一个子系统链接的结构，Simulink 就会提示用户禁止链接。结构更改包括添加或删除一个模块、线，改变模块的端口数等；非结构更改包括改变不影响子系统结构的参数值等。

编者提示：Simulink 在一个封装子系统的参数对话框上显示“Parameterized Link”，该子系统的参数对已链接的封装子系统而言不同于库标准模块。用户可以利用此特性快速确定一个库链接是否与库标准链接不同。

5. Find 对话框

用户通过 Find 对话框不仅可以搜索到 Simulink 模型和流程图中满足指定搜索条件的项目，也可以通过此对话框查找注释、模块、信号、状态数以及状态转换等。Find 对话框的调用可从 Simulink 的 Edit 菜单上选择 Find 即可。

6. 模型浏览器(Model Browser)

模型浏览器的工具栏上增加了两个新按钮：

Show Library Links：选择树-节点形式显示库链接；

Look Under Masks：树-节点形式显示封装模块的内容。

7. 单窗口模式(Single Window Mode)

Simulink 4.0 提供了两种打开子系统的模式：

①多窗口模式：Simulink 将每一个子系统打开在一个新的窗口上；

②单窗口模式：Simulink 关闭上级系统(母系统)，打开子系统。

8. 键盘浏览

Simulink 4.0 提供如表 8.1 所示的键盘快捷方式。

表 8.1 键盘快捷键

键	动 作
Tab	选中模块图上的后一个模块
Shift+Tab	选中模块图上的前一个模块
Ctrl+Tab	当模型浏览器被激活后，交替显示浏览器树窗口或图表窗口。
Enter	打开当前选中的子系统
Esc	打开当前子系统的上级系统（母系统）

9. 增强库浏览器

Simulink 4.0 版本的库浏览器做了优化，包括如下一些新的特性：

• 模块不再以树-节点形式显示，而是以图标的形式显示在预览窗口上。

• 预览窗口已从库树结构窗口的下方移到旁边。用户可以通过拖拉和释放将显示在预览窗口上的模块样本创建在某个模型上。

• 浏览器的窗口被分成两个，可以任意调整大小。

• 双击某个模块的图标即可打开模块参数对话框，但所有参数栏均被禁止。这样用户可以查看，但不能更改库模块的参数。

• 双击某个库模块即可打开预览窗口上的库。

• 用户现在可以通过双击预览窗口上的模块，在其屏幕的最高级模型上插入一个模块，并从接下来出现的菜单上选中 Insert in...。若没有打开的模型，或最高级的模型是一个被锁定的库，则库浏览器提供构建插入模块的模型。

• Simulink 4.0 版本的浏览器包含一个带有 File，Edit 和 Help 选项的菜单。

• 模块的帮助文本窗口已从库浏览器的底部移至顶部。

• 从库浏览器的 Edit 菜单选中 Find 即可显示一个非模态 Find 对话框。

• 浏览器的搜索速度更快，并支持常规表达式。

10. 帮助菜单(Help Menus)

Simulink 4.0 在模型的菜单栏和库窗口上增加了一个 Help 菜单。在一个模块的上下文菜单上的帮助项显示该模块的帮助页。

11. 撤消键入

Simulink 4.1 版本的撤消键入(Undo)命令新增了对模块、注释、连线、节点操作的恢复，同时也可对组成子系统的模块进行恢复。

12. 自动连接

Simulink 4.1 在模块与模块、画信号线等方面又有进一步的改进。如一个模块与一个模块，一组模块与一个模块以及一个模块与多个模块之间的连接更加方便。比如选中源模块，按下 Ctrl 键的同时单击目标模块，系统会自动连接；在 add_line 命令中加入'autorouting'选项可自动连接插入的模块。命令用法请参见第 2 章中相关内容。

13. 兼容性

Simulink 4.1(及以后版本)可以 Simulink 3.0 的格式保存 Simulink 4.0 及以后版本模型。这为 Simulink 3.0 版本的用户提供了极大的便利条件。

从 Simulink File 菜单上选 Save as 选项即可;也可以使用命令:slsaveas(SYS)。

8.2.2 建模改进

1. 层次变量管辖(Hierarchical Variable Scoping)

Simulink 4.0 版本扩充了 Simulink 解决访问封装子系统变量的能力。以前版本的 Simulink 只能解决访问某一个模块的局部工作空间中的变量。在 Simulink 4.0 版本中,用户可以访问位于包含某模块的任意工作空间层次内的变量。

编者提示:在某些情况下,层次管辖会使某些模型的运行在新版本与老版本上有所不同。

2. 矩阵信号

很多新版本的 Simulink 模块接受或输出矩阵信号。一个矩阵信号是用一个矩阵表示的两维信号数组。每一个矩阵元素表示在当前时间步上对应的信号元素的值。除了矩阵信号外,Simulink 也支持标量信号和矢量信号(一维信号元素数组)。Simulink 可以随意地加粗(选中 Format 菜单上的 Wide Lines),并显示携带矢量或矩阵信号的线的维度(选中 Format 菜单上的 Line Dimensions)。当用户选择了 Line Dimensions 选项后,Simulink 会在矩阵信号线的上方显示形式为[r%c]的标签,其中 r 为行数;c 为列数。例如,标签[2%3]表示该线携带了一个 2 行%3 列的矩阵信号。

用户可以使用 Simulink 的源模块,如一个 Sine Wave 或一个 Constant 模块,产生矩阵信号。例如,要创建一个时变矩阵信号,先将 Constant 模块插入模型中,然后将 Constant Value 参数设置为任意一个 MATLAB 赋值矩阵,如[1 2 3 4]。

3. Simulink 数据对象

Simulink 4.0 版本允许模型通过 Simulink 数据对象捕获用户定义的参数和信号信息,例如最小和最大值、单位等等。

4. 模块执行顺序(Block Execution Order)

Simulink 4.0 可随意地显示模型的模块图上的每个模块的执行顺序。

5. 自定义参数

Simulink 4.1 中的命令 add_param 和 delete_param 可以对模块加入自定义参数。

6. ConnectionCallback 参数

Simulink 4.1 在 set_param 命令中增加了 ConnectionCallback 参数用于在连接端口实现函数回调。

7. 绝对误差

Simullink4.1 在 State-Space 模块、Transfer Fcn 模块和 Zero-Pole 模块的参数对话框上均增加了用于指定连续系统的绝对误差状态的选项。

8.2.3　Simulink 调试器

这里仅给出 Simulink 4.0 调试器的新增功能。关于 Simulink 3.0 调试器请参看第 4 章。

1. GUI 调试器接口

与 Simulink 3.0 版本不同的是，Simulink 4.0 在 Simulink 调试器中引入了一个图形用户界面(GUI)。这使得调试过程更为方便、直观。

(1) 启动调试器

打开需要调试的模型，从 Simulink 的 Tools 菜单上选择 Debugger，就可启动调试器。调试器窗口如图 8.3 所示。

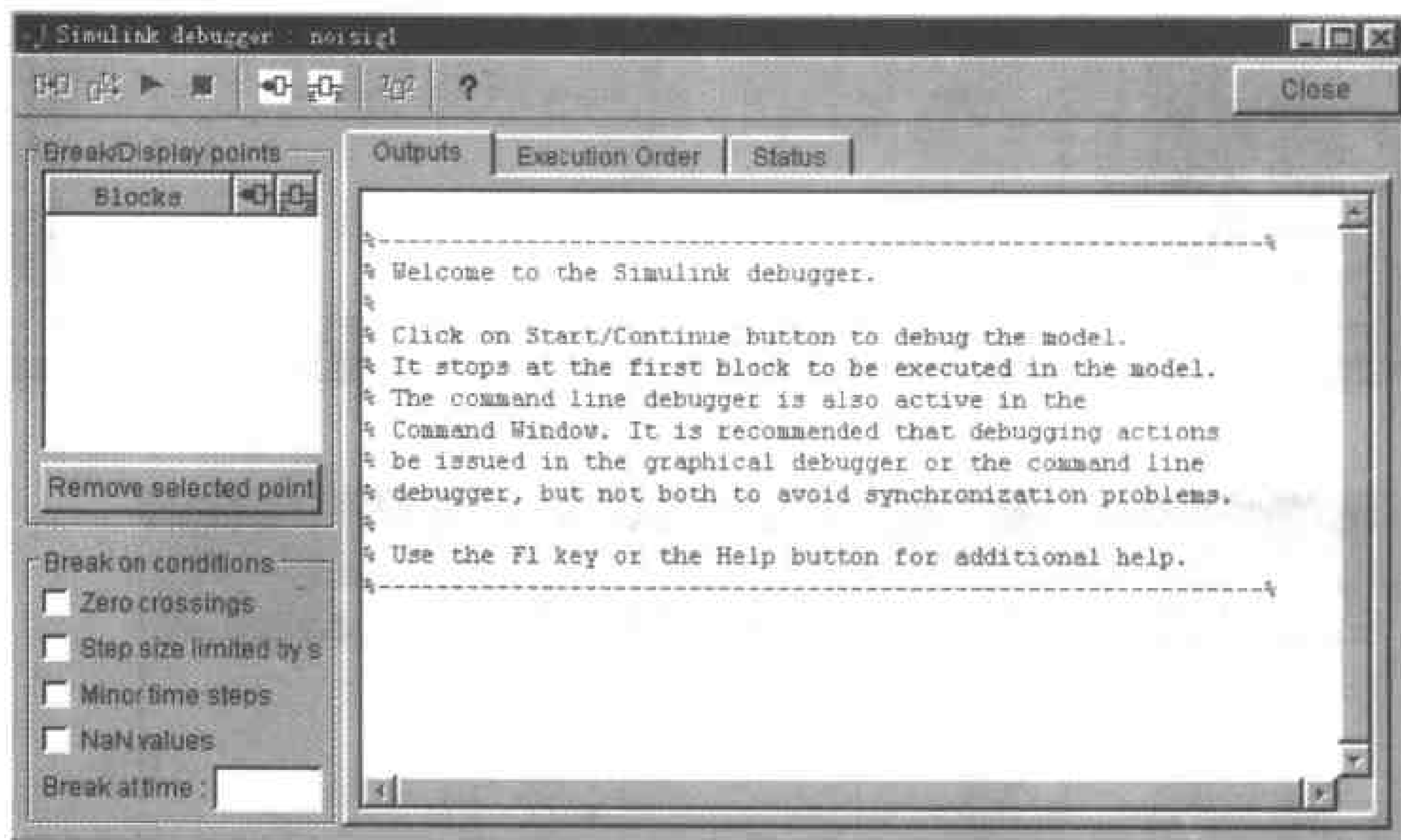


图 8.3　Simulink 4.0 版本调试器窗口

也可以在 MATLAB 命令行输入 sldebug 命令或 sim 命令的 debug 选项来启动处于调试器控制下的模型。例如，输入下列命令：

```
>>sim('vdp',[0,10], simset('debug','on'))
```

或命令

```
>>sldebug 'vdp'
```

将把 Simulink 演示程序模型 vdp 装入存储器，并启动仿真，在模型执行列表的第一个模块上停止仿真。

编者提示：必须确切启动仿真，才能使调试器以 GUI 模式运行。

(2) 启动仿真

选择调试器的工具栏上的开始/继续按钮即可启动仿真，如图 8.4 所示。

仿真开始并停止在执行的第一个模块上。调试器打开模型窗口的浏览器窗口，并高亮显示在模型执行停止的模块上(图 8.5)。

如果调试器是通过命令启动的，则不仅调试器显示仿真时间，而且如同第 4 章一样，在 MATLAB 命令窗口内也会显示出调试命令的提示符；而当调试器以 GUI 模式运行时，则显

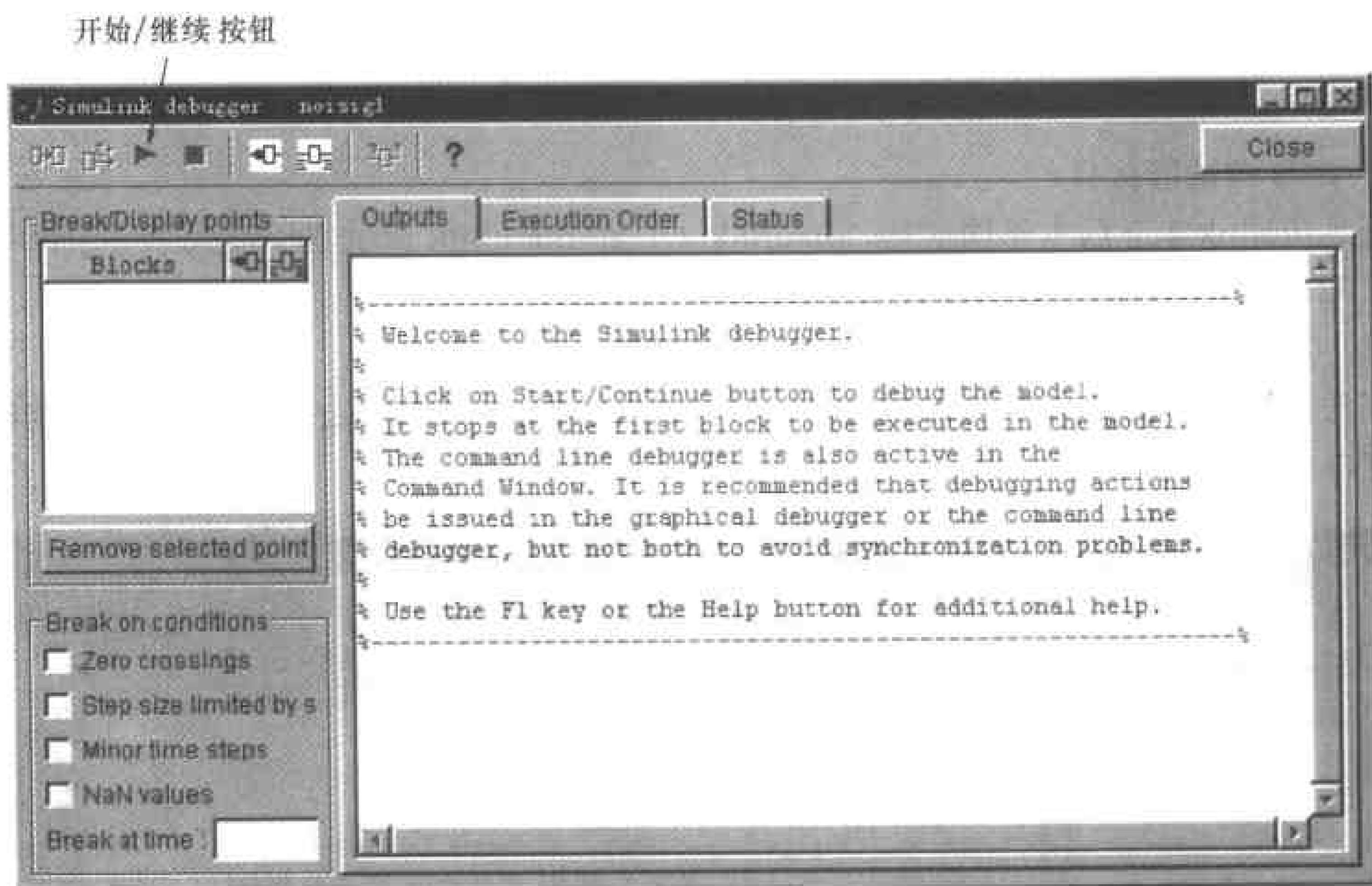


图 8.4　启动/继续仿真的按钮

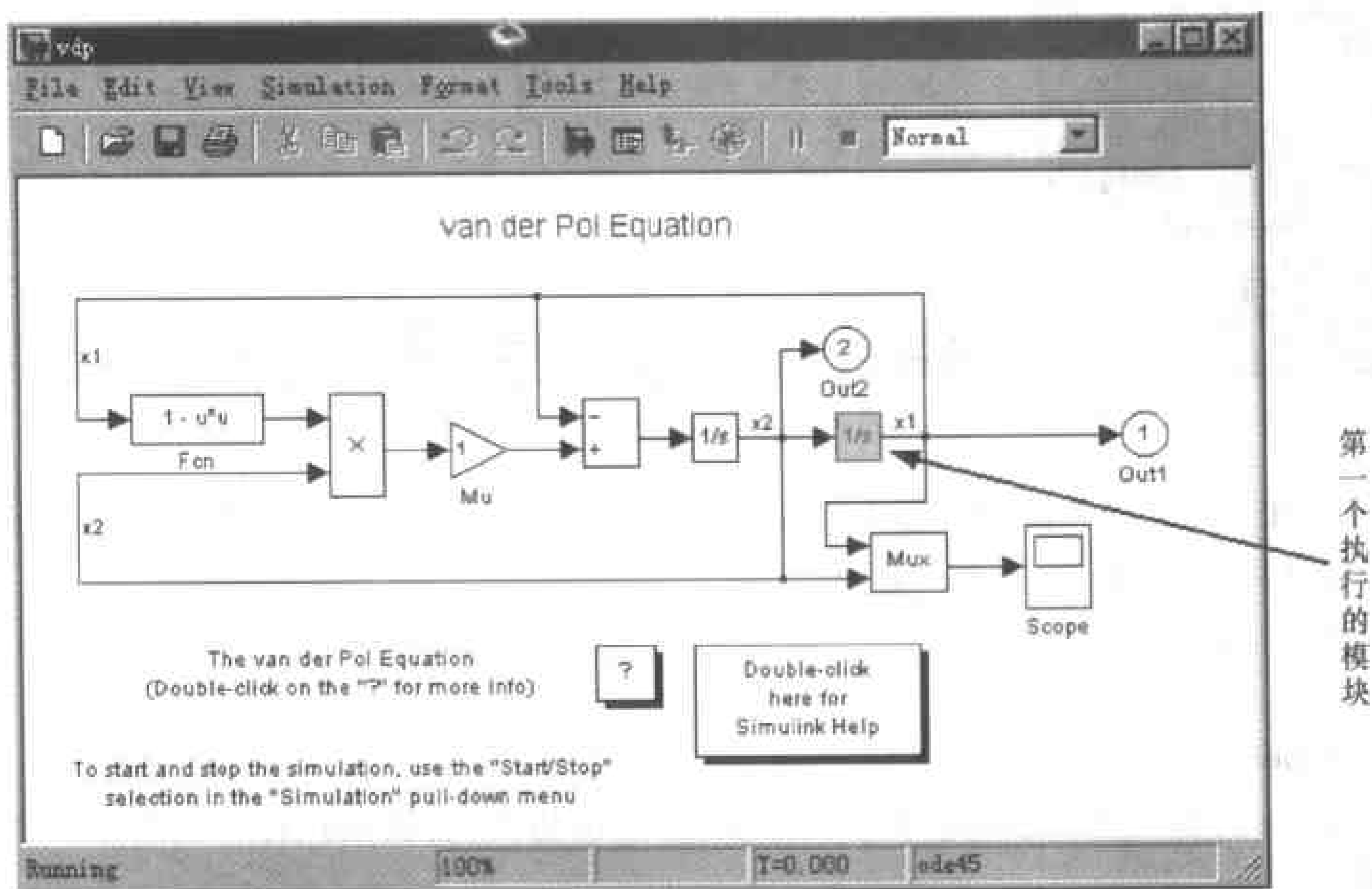


图 8.5　启动仿真后的第一次暂停

示在调试器的输出窗口内(图 8.6)。

命令提示显示模块指数和第一个执行的模块的名。

编者提示:当用户以 GUI 模式启动调试器后,调试器的命令行界面在 MATLAB 命令窗口中也是激活的。所以应避免使用命令行界面以避免图形界面和命令行界面可能发生的同步错误。

关于使用调试器的命令行界面,可参看第 4 章“Simulink 仿真模型动态调试器”中的有关内容。



图 8.6 GUI 模式下，仿真时间的显示举例

(3) 获得联机帮助

获得调试器联机帮助(图 8.7)的方法：

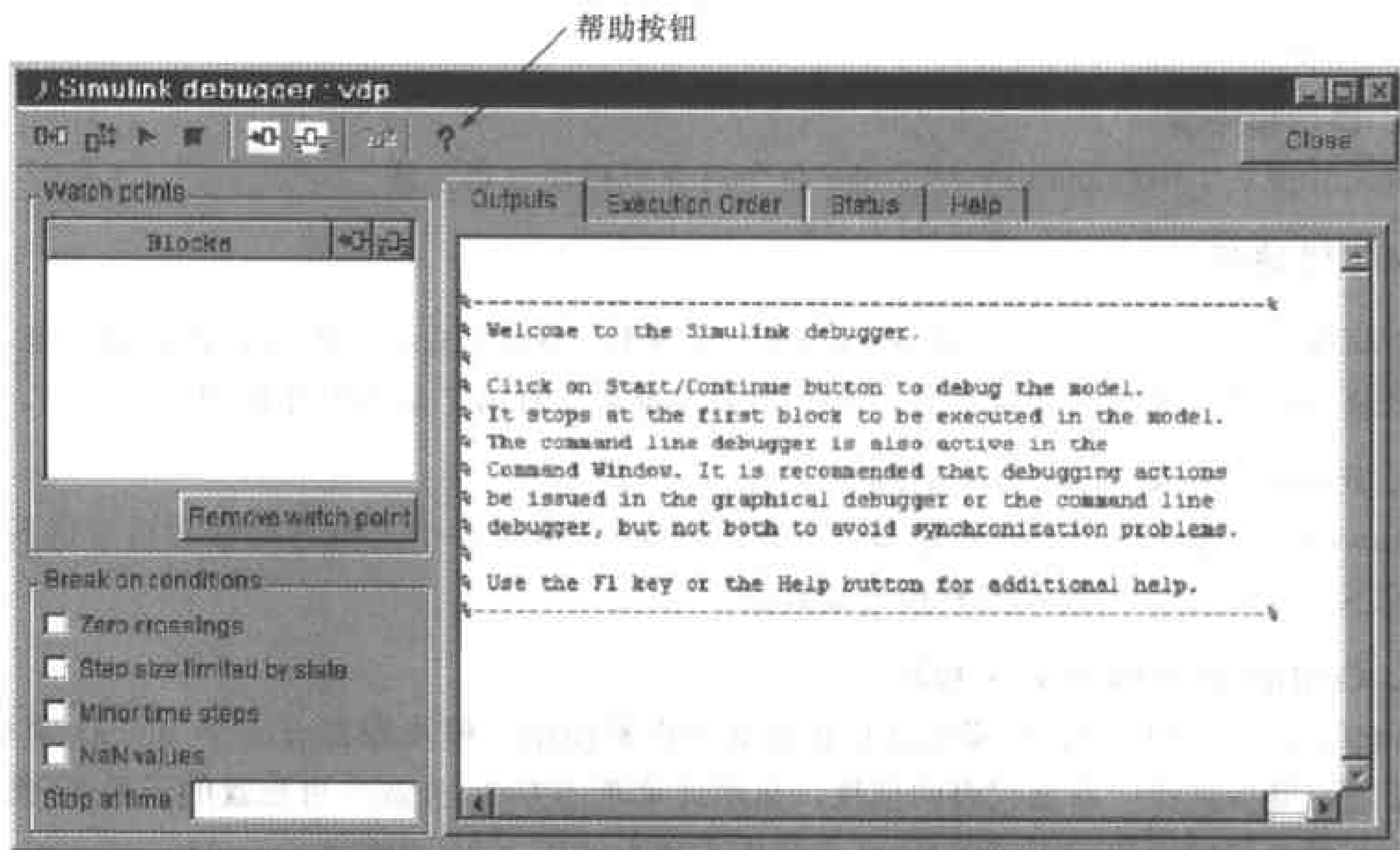


图 8.7 帮助按钮示意

① 可以通过选中调试器的工具栏上的 Help 按钮，在 MATLAB 帮助浏览器内显示调试器的帮助；

② 当文本光标处于一个调试器窗口或文本栏中时，按 F1 键均可使用联机帮助。按下 F1

键显示调试器窗口或当前为键盘输入点的文本栏的帮助。在命令行模式下，用户也可以通过在调试提示符后键入 help 获得调试器命令的简要描述。

(4) 运行仿真

在 GUI 模式下，通过选择调试器工具栏上的相应按钮(图 8.8)，用户选择仿真的步进量。

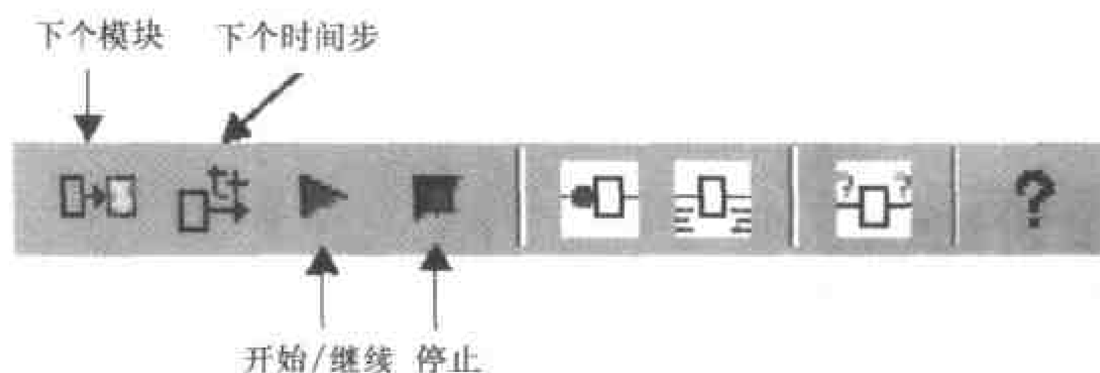


图 8.8 GUI 模式下，设置仿真时间及参量按钮

至于在命令行模式下的选择，可参看第 4 章的相关内容。

2. slist 命令

Simulink 4.1 版本的 slist 命令也显示系统函数模块中可能的系统函数名。

3. 加速器模式

Simulink 4.1 中的 trace、break、zcbreak、nanbreak 和 minor 命令显示，这些命令不适用于加速器模式的信息。

4. 显示和探测

Simulink 4.1 中的 display 和 probe 命令对虚拟模块也可工作。

8.2.4 模块库

下面按照英文字母排序，介绍 Simulink 4.0 版本模块库(Block Library)的新增功能、新增模块以及某些已有模块的功能扩展。其他相关内容请参看第 7 章中对相应模块的描述。

1. Bitwise Logical Operator 模块

Bitwise Logical Operator 模块是一个新的模块，它可对一个无符号的整数信号进行逻辑屏蔽(AND、OR、XOR)，反相或移位等运算。

2. Configurable Subsystem 模块

Configurable Subsystem 模块已重新编制而更易使用。模块现在有一个 Blocks 菜单，用户可通过该菜单选择子系统代表的模块。显示此菜单的方法：先选中可配置的子系统，然后从 Simulink 编辑器的菜单或右击鼠标，在出现的菜单上选中 Edit 即可。

3. Function Call Generator 模块

Simulink 4.0 版本的 Function Call Generator 模块新增了 Number of iterations(函数反复调用)参数。用户可通过该参数指定每一个时间步内调用目标模块的次数。

4. Gain 模块

Gain 模块现在支持用一个增益因子对输入矩阵和元素法则的乘。模块对话框新增一个 Multiplication 参数以便用户对如下选项进行选择：

- K * u（元素法则乘）
- K * u（增益左乘）
- u * K（增益右乘）

5. LUT 模块

新版本提供了四个新的 Look-Up Table（LUT）模块：

- Direct Look-Up Table（n-D）
- Look-Up Table（n-D）
- Prelook-Up Index Search
- Interpolation（n-D）Using PreLook-Up

这些模块均位于 Simulink 的 Functions and Tables 模块库中。

6. Math Function 模块

Math Function 模块新增两个特殊矩阵函数：矩阵的转置和矩阵的厄密(Hermitian)共轭。前者的输出为输入矩阵的转置；后者的输出为输入矩阵的复共轭转置。

7. Polynomial 模块

（1）功能描述

新增 Polynomial 模块位于 Functions and Tables 库中。模块使用系数参数将输入值赋值给多项式。用户以 MATLAB 的 polyval 命令可接受的形式定义一组多项式系数。模块将在每一个时间步内的 U 计算 P(u)。

输入和系数必须是双精度型或单精度型的实信号。Polynomial coefficients 参数必须与输入的数据类型相同。输出数据类型同输入。

（2）参数与对话框

Polynomial 模块的参数对话框如图 8.9 所示。

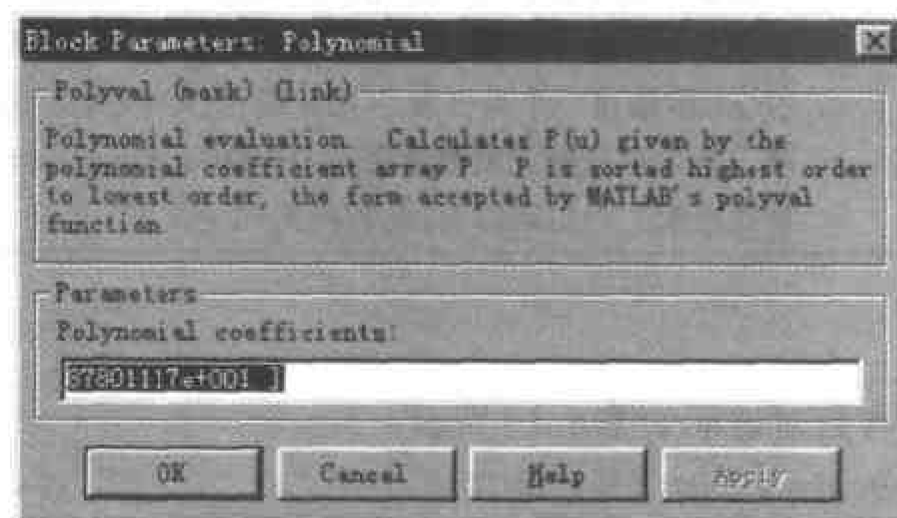


图 8.9 Polynomial 模块的参数对话框

- Polynomial coefficients：MATLAB polyval 形式的多项式系数的值，第一个为 x^N 的系数，第二个为 x^{N-1} 的系数，……，最后一个是常数。详见 polyval 命令。

(3) 特性

直通输出;采样时间从驱动模块继承;可进行维度定制。

8. Probe 模块

Probe 模块现在可即时输出模块输入信号的维度。

9. Product 模块

Product 模块现在支持输入的元素%元素、矩阵乘以及矩阵求逆。模块的对话框包括一个新增参数 Multiplication,通过此参数,用户可以指定模块是对输入进行元素%元素或矩阵%矩阵间的乘或求逆。

10. Reshape 模块

Simulink 4.0 新增了 Reshape 模块(在 Signals & Systems 库中),通过指定 Output dimensionality 参数可改变输入信号的维度,例如,本模块可将 n 元素的矢量变换为 $1\times N$ 或 $N\times 1$ 的矩阵信号,反之亦然。模块接受和输出任意数据类型的信号。

(1) 功能描述

Output dimensionality 参数让用户选择表 8.2 所列的任意输出选项。

表 8.2 Reshape 模块的 Output dimensionality 参数

Output dimensionality 输出维度	描 述
1-D array (一维度列)	将矩阵(二维度列)转换成矢量(一维度列)数列信号。输出矢量由矩阵的第一列,后跟矩阵的第二列等构成。本选项不改变矢量输入
Column vector (列矢量)	将一个矢量或矩阵输入信号转换成一个列矢量。例如 M%1 的矩阵,其中 M 为输入信号的元素的数量。对矩阵而言,此转换为列主次序
Row vector (行矢量)	将一个矢量或矩阵输入信号转换成一个行矩阵,即 1%N 矩阵,其中 N 为输入信号中元素的数量。对矩阵而言,此转换为列主次序
Customize (定制化)	将输入信号转换成用户使用 Output dimensions 参数定义维度的输出信号。Output dimensions 参数的值可以是一个或两个元素矢量。[N]表示输出一个长度为 N 的矢量。[M N]表示输出一个 M%N 的矩阵。输入信号的元素数量必须与 Output dimensions 参数指定的元素数量相匹配。对矩阵而言,此转换为列主次序

(2) 参数和对话框

- Reshape 模块的参数对话框如图 8.10 所示。
- Output dimensionality:输出信号的维度。
- Output dimensions:指定一个输出维度。此选项只有当用户将 Customize 指定为 Output dimensionality 参数的值时才被激活。

(3) 特性

直通输出;采样时间从驱动程序继承;可进行维度定制。

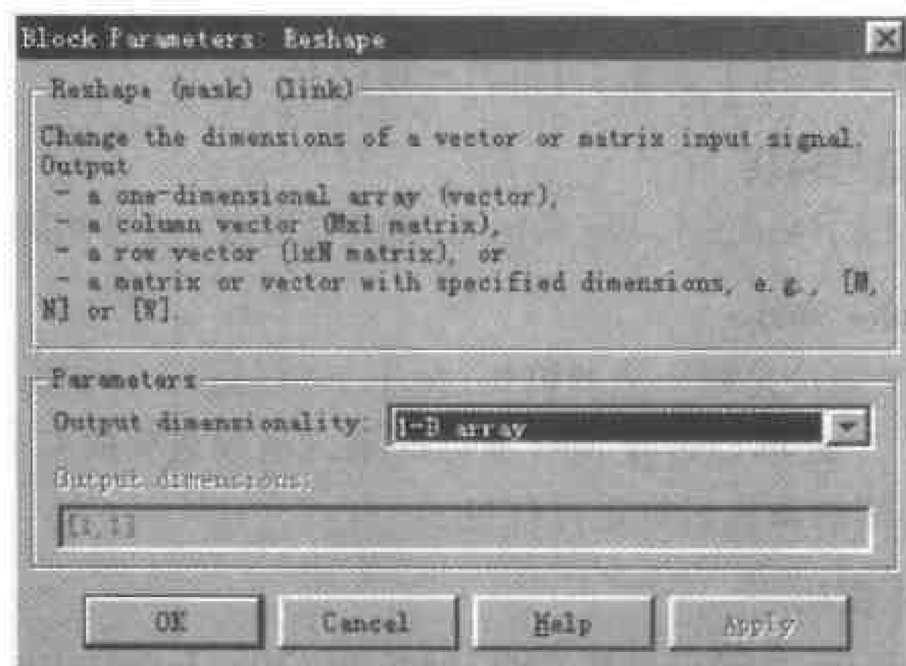


图 8.10　Reshape 模块的参数对话框

编者提示：Simulink 4.X 中已将以前版本中的矢量化(Vectorized)特性变更为维度定制(Dimensionalized)特性。该参量表明模块可操作多维度数据。请 4.X 版本的用户在阅读本书时注意。

11. Signal Specification 模块

(1) 功能描述

新增 Signal Specification 模块位于 Signals & Systems 库中，用户可通过模块指定输入信号必须满足的属性，包括维度、采样时间、信号的数据和数值类型。模块检验输入是否满足用户指定的属性，如果满足则输出，否则暂停仿真，并给出出错信息。模块接受任意类型的信号，只要信号的数据类型与常数指定的类型相匹配。

(2) 参数与对话框

Signal Specification 模块的参数对话框如图 8.11 所示。

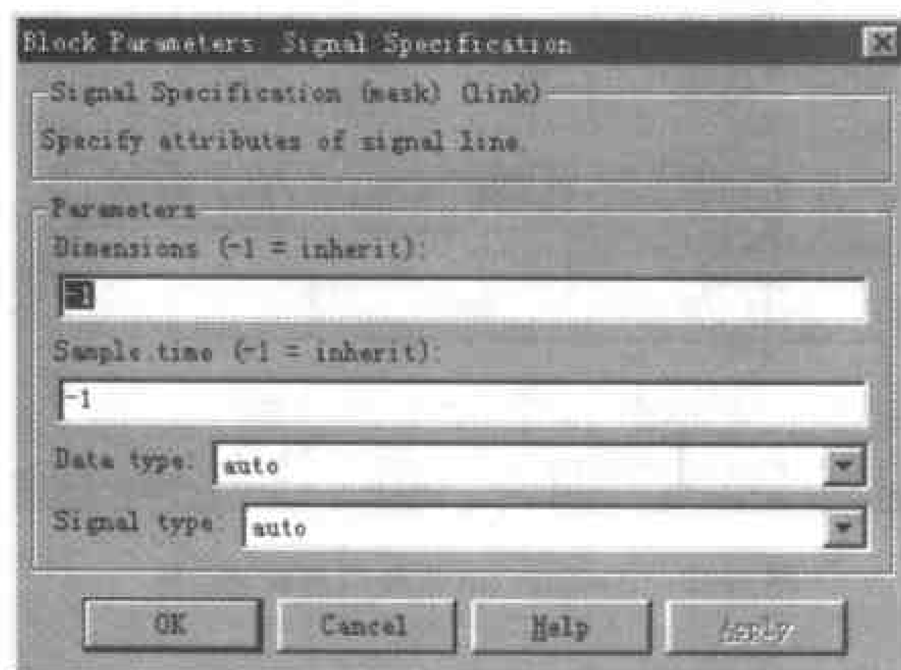


图 8.11　Signal Specification 模块的参数对话框

参数包括 Dimensions，Sample Time，Data Type 和 Signal Type。

（3）特性

直通输出；连续采样时间；0状态数；可进行维度定制。

12. 多路传输矩阵信号模块

Simulink 4.X中的Mux、Demux和Bus Selector模块已增强为支持矩阵信号的多路传输。

13. S-function Builder 模块

Simulink 4.1新增了系统函数生成器模块（位于Functions & Tables库中），用户可以通过对话框输入指定参数，生成其C语言Mes文件的系统函数或生成C语言源代码，另外可以对通过Simulink模型创建的系统函数进行打包。

14. Pulse Generator 模块

在Simulink 4.1版本中，已将以前独立的Discrete Pulse Generator模块归入Pulse Generator模块，并随之增加了Pulse type参数用于指定连续或离散信号。

15. 条件执行模块

Simulink 4.1新增了几种条件执行模块：If、Switch Case与Action Port模块，以及For Iterator和While Iterator模块。这些模块均在Subsystems库中。

16. Assigment 模块

这是4.1版本新增的模块，位于Signals & Systems库中。

（1）功能描述

用于将指定的值赋予指定的标量、矢量或矩阵信号元素。输入及输出信号类型均为双精度型。

（2）参数及对话框

Assignment模块的参数对话框如图8.12所示。

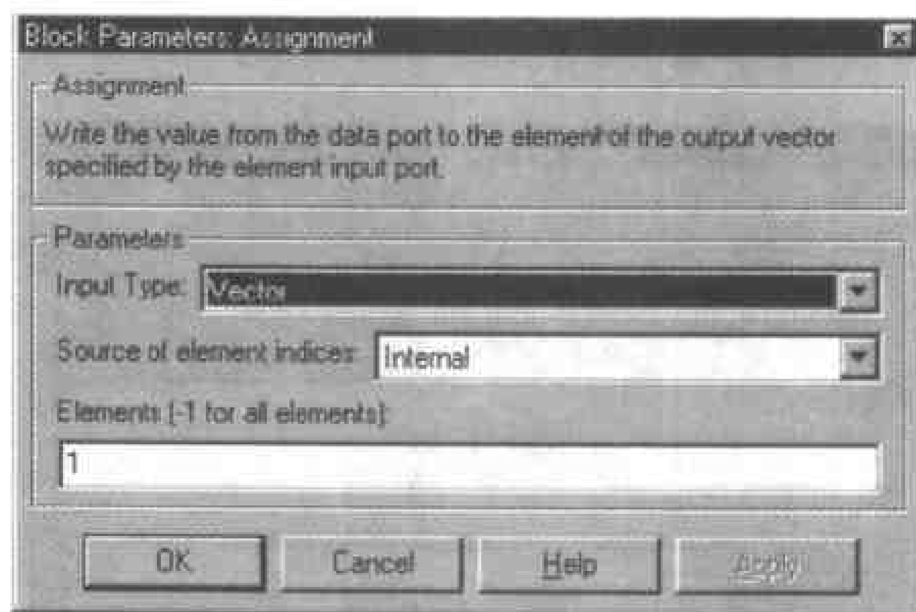


图8.12　Assignment模块的参数对话框

编者提示：若在Input Type栏内选择了Matrix，则显示Source of row indices和Source of column indices栏。

17. Bus Creator 模块

Simulink 4.1新增的模块，位于Signals & Systems库。

(1) 功能描述:将多个信号线组合成一个总线。

(2) 参数与对话框:

Bus Creator 模块的参数对话框如图 8.13 所示。



图 8.13 Bus Creator 模块的参数对话框

18. Scope 模块

Simulink 4.1 版本的 Scope 模块又进行了以下的改进:

(1) Scope 模块的浮动模式现独立为 Floating Scope 模块;

(2) Floating Scope 模块保存模型文件选择显示的信号;

(3) Scope 模块的工具箱上有浮动和非浮动间的转换按钮,可保存 saved axes,locking/unlocking axes 以及显示 Signal Selector。

19. 系统功能扩展

(1) ADA S-Functions

Simulink 4.X 支持以 ADA 编码的 S-functions。

(2) 基本子系统(Atomic Subsystems)

Simulink 4.X 允许用户将子系统指派为与 virtual 对立的 atomic。一个基本子系统是一个真实的子系统。在仿真一个模型时,Simulink 首先执行所有包含在基本子系统中的模块,然后执行包含模型(或基本子系统)的模块。

通过声明一个子系统为基本子系统,用户可以保证 Simulink 在基本子系统执行完成后再执行其他处于模型同一层(级)的模块。

编者提示:条件执行子系统都是固有的 atomic。Simulink 不允许用户指定它们是 atomic 或 virtual。

8.2.5　SB2SL

SB2SL是Simulink的组成部分之一，它允许用户将SystemBuild SuperBlocks编译成Simulink模型。

增强的SB2SL 2.1新版本为Real-Time Workshop的使用提供了更完善的支持。若用户使用Real-Time Workshop 4.0生成模型(已使用SB2SL将SystemBuild转换为Simulink)的代码，那么将生成模型中的大多数已转化的模块的代码。

本程序不支持通过如下组件进行代码生成：

- ConditionBlock
- Decoder
- Encoder
- GainScheduler
- Interp Table (Archive library)
- ShiftRegister

SB2SL 2.1也包括一些重要的错误调整组件。

8.3　Simulink 4.X的运行工具

8.3.1　Simulink 4.0的运行工具简介

运行工具(Performance Tools)是从Simulink 4.0版本开始新增的功能，运行工具包括：

- Simulink Accelerator(加速器)
- Model Differencing Tool(模型差异工具)
- Profiler(运行简档)
- Model Coverage Tool(模型覆盖率工具)

编者提示：要使用这些工具，必须事先装入。

8.3.2　Simulink加速器

Simulink加速器用于提高Simulink模型的执行速度。加速器使用Real-Time Workshop(可将Simulink模型自动生成C语言代码)的某些部分，并将用户的C语言编译程序创建成一个可执行程序。需要注意的是，虽然Simulink加速器利用了Real-Time Workshop技术的优点，但Real-Time Workshop并不是必须运行这个加速器。也就是说，若Windows PC用户没有预装C语言编译器，则可以使用MathWorks提供的Icc编译器。

另外，要使用加速器，必须将Simulink Performance Tools选项装入系统。运行Simulink加速器的方法是：在打开的用户模型上，从Simulation菜单上选中Accelerator，就可打开Simulink加速器。如图8.14所示的模型。

具体使用方法详见新版的Simulink相关资料。



图 8.14　F14 飞机控制系统模型

8.3.3　模型差异工具

模型差异工具(Model Differencing Tool)寻找并显示两个 Simulink 模型之间的差异。这使得用户可以快速地确定两个模型的不同,如相同模型的不同版本等。

另外,使用前,必须将 Simulink Performance Tools 选项装入系统。

要使用本工具,先打开欲比较的两个模型之一,然后选择 Simulink Tools 菜单上 Model differences 选项。Model Differencing Tool 随同 Select Second Model(选择第二个模型)对话框一起出现。如图 8.15 所示。

用 Select Second Model 对话框选择欲比较的另一个模型。若需要,可打开第二个模型,两个模型将自动被安排(图 8.16)。

模型差异工具有三个窗口。左上角的窗口显示第一个模型的内容(可扩展表)。右上角的显示第二个模型的内容(可扩展表)。颜色表示两个模型之间的差别。

- 蓝色标注在两个模型之一中出现的模块。
- 红色标注在两个模型中出现但参数值或内容不同的模块。
- 绿色标注在两个模型中完全系统相同的模块。

单击窗口内的某个模块就使模型窗口中的相应模块图标高亮显示。底部的窗口显示存在于两个模型中已选中的模块的不同版本之间的参数差别。



图 8.15　选择第二个模型对话框

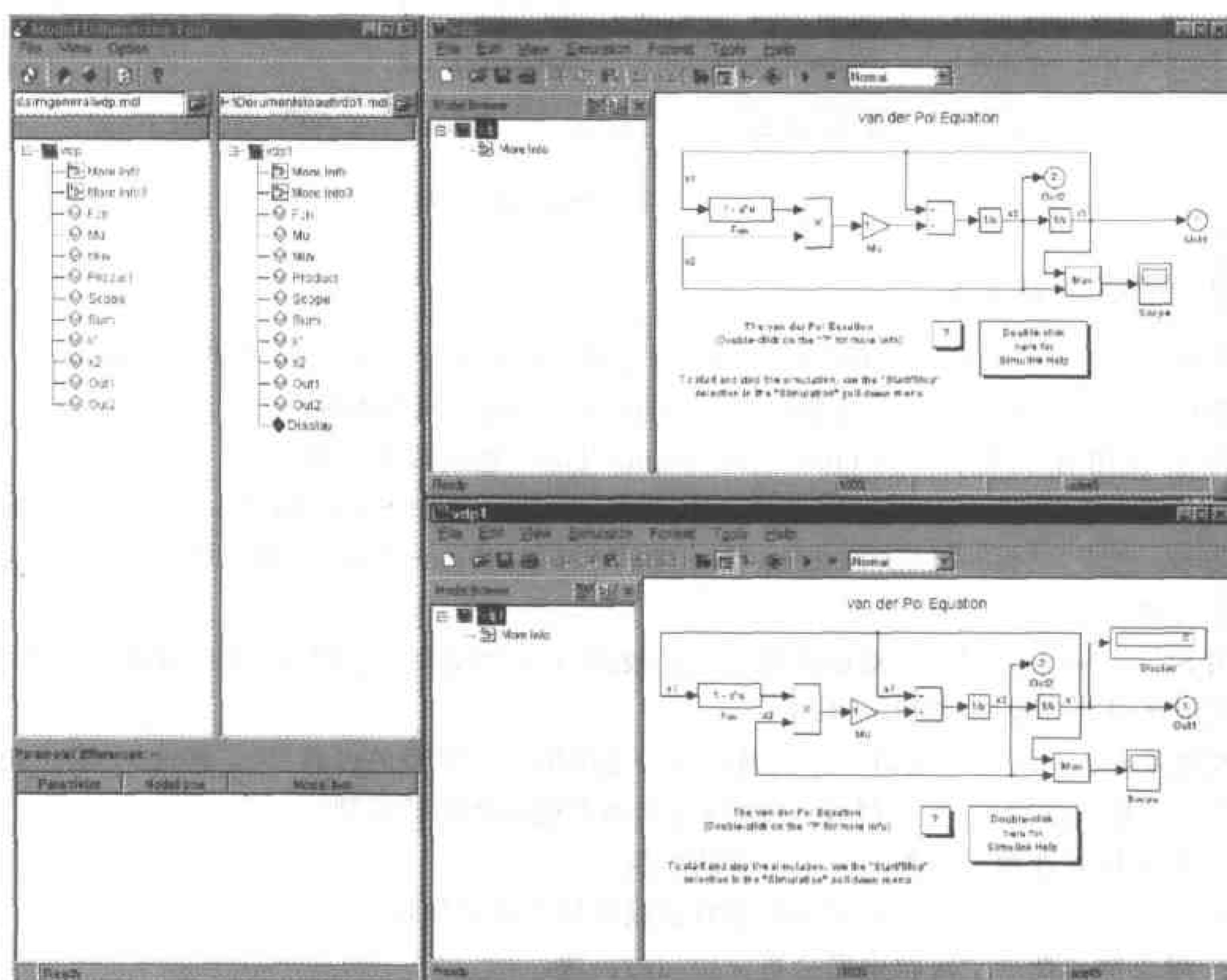


图 8.16　模型差异窗口(含两个模型)

8.3.4 运行档案器

Simulink 仿真运行档案在用户模型的仿真期间收集运行数据，并给出数据报告(仿真档案)。由档案器(Profiler)产生的仿真档案，显示出 Simulink 执行每一个用于用户模型仿真的功能所花费的时间。用户通过仿真档案能够确定自己模型的某个部分需要仿真的时间和实施模型最优化的位置。

另外，使用前，必须将 Simulink Performance Tools 选项装入系统。

具体使用方法详见新版 MATLAB 6.X 资料。

8.3.5 模型覆盖率工具

模型覆盖率工具(Model Coverage Tool)决定一个模型测试事件历经模型的仿真路径的范围。一个测试事件历经路径的百分比被称为模型覆盖率。模型覆盖率是一种衡量模型测试的彻底性的方法。因此模型覆盖率工具可以帮助用户验证用户模型的测试。

另外，使用前，必须从 MATLAB 主窗口上左侧的 Launch Pad 内将 Simulink Performance Tools 选项装入系统。

具体使用方法详见 MATLAB 6.X 的相关资料。

编者提示：总体上看，Simulink 4.1 对以前版本中存在的一些缺陷进行了一些修补，如：基本子系统中的变化采样时间系统函数不再会崩溃；Bus Selector 输入总线名多重探测；某些模块在仿真期间的存储器优化；运行档案留存(见 tempdir 命令)；输入源库的信号发生器算法改进等等。据编者了解，目前 MATLAB 的最新版本是 6.5 版，可以肯定，它还会继续“升级”。但如同所有软件一样，并非最新的版本就一定最好。读者应根据需要来选择其版本。如果是新用户，自然应该选用最新版本；若是老用户，也不必担心，因为熟练、快速、适用才是根本。

附录 模型、模块参数及模型文件格式

本附录主要包括两个部分：一是给出模型、模块和封装的参数表，用户可根据参数表所提供的详细信息，通过 set_param 命令（详见第 5 章）修改模型；二是给出了模型文件的格式，以便用户在创建自己的应用程序时使用。

编者提示：本附录只适用于 Simulink 3.0 版本。

A. 模型参数

附表 1 列出并描述了一个模型的参数。参数的出现顺序按参数在模型文件中的定义顺序。该表也包含模型的调回参数（见第 3 章"使用调回例程"）。描述列指出用户在 Simulink Parameters 对话框上设置数值的位置。模型参数即仿真参数在第 4 章中已做了详细的描述。表 1 下面的例子说明更改参数的方法。

参数值必须指定为用引号括入的字符串的格式。字符串的内容取决于参数，可以是数值[标量(scalar)、矢量(vector)或矩阵(matrix)]、一个变量(variable)名、一个文件名或是一个具体值。值列指出要求的数值类型、可能的值（用"|"隔开）以及括号内的默认值。

附表 1 模型参数

参数	描述	值
Name	模型名	文本
Version	用于修改模型的 Simulink 版本（只读）	(release)
SimParamPage	Simulation Parameters 对话框显示页（最后一个显示页）	{Solver} | WorkspaceI/O |Diagnostics
SampleTimeColors	Sample Time Colors 菜单选项	on | {off}
InvariantConstants	不变常数设置	on | {off}
WideVectorLines	Wide Vector Lines 菜单选项	on | {off}
ShowLineWidths	Show Line Widths 菜单选项	on | {off}
PaperOrientation	打印纸方向	portrait | {landscape}
PaperPosition	图表在纸上的位置	[left, bottom, width, height]
PaperPositionMode	纸位模式	auto | {manual}
PaperSize	PaperUnits 中 PaperType 的尺寸	[width height] (read only)

续附表 1

参数	描述	值
PaperType	打印纸型	{usletter} | uslegal | a0 | a1 | a2 | a3 | a4 | a5 | b0 | b1 | b2 | b3 | b4 | b5 | arch-A | arch-B | arch-C | arch-D | arch-E | A | B | C | D | E | tabloid
PaperUnits	打印纸尺寸单位	normalized | {inches} | centimeters | points
StartTime	仿真开始时间	标量 {0.0}
StopTime	仿真终止时间	标量 {10.0}
Solver	解算器(仿真器)	{ode45} | ode23 | ode113 | ode15s | ode23s | ode5 | ode4 | ode3 | ode2 | ode1 | FixedStepDiscrete | VariableStepDiscrete
RelTol	相对误差容限	标量 {1e−3}
AbsTol	绝对误差容限	标量 {1e−6}
Refine	改善因子	标量 {1}
MaxStep	最大步长	标量 {auto}
InitialStep	初始步长	标量 {auto}
FixedStep	固定步长	标量 {auto}
MaxOrder	ode15s 的最高阶数	1 | 2 | 3 | 4 | {5}
OutputOption	输出选项	AdditionalOutputTimes | {RefineOutputTimes} | SpecifiedOutputTimes
OutputTimes	OutputOption 值	矢量 {[]}
LoadExternalInput	从工作空间加载输入	on | {off}
ExternalInput	时间和输入变量名	标量或矢量 [t, u]
SaveTime	保存仿真时间	{on} | off
TimeSaveName	仿真时间名	变量{tout}
SaveState	保存状态	on | {off}
StateSaveName	状态输出名	变量 {xout}
SaveOutput	保存仿真输出	{on} | off
OutputSaveName	仿真输出名	变量{yout}
LoadInitialState	加载初始状态	on | {off}
InitialState	初始状态名或值	变量或矢量 {xInitial}
SaveFinalState	保存最终状态	on | {off}

续附表 1

参数	描述	值
FinalStateName	最终状态名	变量 {xFinal}
LimitMaxRows	限定输出	on | {off}
MaxRows	要保存的最大输出行数	标量 {1000}
Decimation	抽取因子	标量 {1}
AlgebraicLoopMsg	代数环鉴别	none | {warning} | error
MinStepSizeMsg	最小步长鉴别	{warning} | error
UnconnectedInputMsg	未接输入端口鉴别	none {warning} | error
UnconnectedOutput-Msg	未接输出端口鉴别	none | {warning} | error
UnconnectedLineMsg	未接连线鉴别	none | {warning} error
ConsistencyChecking	一致性验证	on | {off}
ZeroCross	固有过零检测(见第9章"过零检测")	{on} | off
CloseFcn	关闭调回	命令或变量
PreLoadFcn	预载调回	命令或变量
PostLoadFcn	后载调回	命令或变量
SaveFcn	保存调回	命令或变量
StartFcn	开始仿真调回	命令或变量
StopFcn	终止仿真调回	命令或变量
BooleanDataType	使能布尔模式	on | {off}
BufferReuse	使能重新使用模块I/O缓冲器	{on} | off

下面这些例子给出设置 mymodel 系统的模型参数的方法。

例1:下面的命令设置仿真的开始和终止时间:

```
>>set_param('mymodel','StartTime','5','StopTime','100')
```

例2:下面的命令将解算器设置为ode15s,并更改最大阶数:

```
>>set_param('mymodel','Solver','ode15s','MaxOrder','3')
```

例3:下面的命令发生一个SaveFcn调回:

```
>>set_param('mymodel','SaveFcn','my_save_cb')
```

B. 模块的共有参数

附表2列出包括模块调回参数(详见第3章"使用调回例程")在内的所有 Simulink 模块的共有参数。附表2后给出更改这些参数的实例。

附表 2　共有模型参数

参数	描述	值
Name	模块名	字符串
Type	Simulink 对象类型(只读)	'block'
Parent	拥有该模块的系统名	字符串
BlockType	模块类型	文本
BlockDescription	模块描述	文本
InputPorts	输入端口位置排列	[h1,v1; h2,v2; ...]
OutputPorts	输出端口位置排列	[h1,v1; h2,v2; ...]
CompiledPort-Widths	端口列宽	标量或矢量
ForegroundColor	模块的名、图标、外形、输出信号以及信号标签色	{black} | white | red | green | blue | cyan | magenta | yellow | gray | lightBlue | orange | darkGreen
BackgroundColor	模块图标背景色	black | {white} | red | green | blue | cyan | magenta | yellow | gray | lightBlue | orange | darkGreen
DropShadow	显示阴影	{off} | on
NamePlacement	模块名的位置	{normal} | alternate
FontName	字体	{Helvetica}
FontSize	字体尺寸	{10}
FontWeight	字体加重(取决于系统)	light | {normal} | demi | bold
FontAngle	字体倾斜(取决于系统)	{normal} | italic | oblique
Position	模型窗口中模块位置	矢量 [left top right bottom],不用引号
ShowName	显示模块名	{on} | off
Tag	用户定义字符串	' '
UserData	任意 MATLAB 数据类型(未存入 mdl 文件)	[]
Selected	模块所选状态	on | {off}
CloseFcn	关闭调回	MATLAB 表达式
CopyFcn	复制调回	MATLAB 表达式
DeleteFcn	删除调回	MATLAB 表达式
InitFcn	初始化调回	MATLAB 表达式
LoadFcn	加载调回	MATLAB 表达式
ModelCloseFcn	模型关闭调回	MATLAB 表达式
NameChangeFcn	模块名更改调回	MATLAB 表达式
OpenFcn	打开调回	MATLAB 表达式

续附表 2

参数	描述	值
ParentCloseFcn	Parent 子系统关闭调回	MATLAB 表达式
PreSaveFcn	预存调回	MATLAB 表达式
PostSaveFcn	后存调回	MATLAB 表达式
StartFcn	运行仿真调回	MATLAB 表达式
StopFcn	终止仿真调回	MATLAB 表达式
UndoDeleteFcn	撤消模块删除调回	MATLAB 表达式
LinkStatus	模块的链接状况	none | resolved | unresolved | implicit
AttributesFormatString	指定在模块图中的模块下方要显示的参数	字符串

下面举例说明更改表中参数的方法。

例 1：下面的命令更改方向为从右到左。

```
>>set_param('mymodel/Gain','Orientation','left')
```

例 2：下面的命令使一个 OpenFcn 调回与 mymodel 系统中的 Gain 模块发生关联。

```
>>set_param('mymodel/Gain','OpenFcn','my_open_cb')
```

例 3：下面的命令设置 mymodel 系统中 Gain 模块的 Position 参数。模块宽 75 个像素，高 25 个像素。位置矢量不括人引号。

```
>>set_param('mymodel/Gain','Position',[50 250 125 275])
```

C. 模块的特有参数

附表 3 至附表 10 列出了所有 Simulink 模块的模块特有参数。在使用 set_param 命令设置模块参数时，用户是通过指定模块的 BlockType 参数（出现在模块名后面的圆括号内）来确定模块的。

虽然在以下各表中包含了封装模块的 MaskType 参数值，但所包含的详细信息仅仅是关于内置模块的，而非封装模块。详情请参见“D. 封装参数”。

“对话框提示”列简要说明了在模块的对话框上参数的提示信息的内容。“值”列给出了要求数值类型（标量、矢量、变量）、可能值（用“|”分隔）以及默认值（在括号内）。

附表 3　Sources 库模块参数

模块（模块类型）/参数	对话框提示	值
Band-Limited White Noise (Continuous White Noise)（封装）		
Chirp Signal (chirp)（封装）		
Clock(Clock)（无模块特有参数）		
Constant (Constant) Value	常数值	标量或矢量{1}

续附表 3

模块(模块类型)/参数	对话框提示	值
Digital Clock (DigitalClock) SampleTime	采样时间	标量(采样周期) {1}或矢量[周期偏置]
Digital Pulse Generator		
From File (FromFile)		
FileName	文件名	文件名{untitled. mat}
From Workspace (FromWorkspace)		
VariableName	矩阵表	矩阵{[T,U]}
Pulse Generator(Pulse Generator)(封装)		
Ramp (Ramp)(封装)		
Random Number (RandomNumber) Seed	初始种子	标量或矢量{0}
Repeating Sequence(RepeatTable)(封装)		
Signal Generator (SignalGenerator)		
WaveForm	波形	{sine} | square | sawtooth | random
Amplitude	幅度值	标量或矢量{1}
Frequency	频率	标量或矢量{1}
Units	单位	{Hertz} | rad/s
Sine Wave (Sin)		
Amplitude	幅度值	标量或矢量{1}
Frequency	频率	标量或矢量{1}
Phase	相位	标量或矢量{0}
SampleTime	采样时间	标量(采样周期) {-1} vector [周期偏置]
Step (Step)		
Time	步长	标量或矢量{1}
Before	初始值	标量或矢量{0}
After	最终值	标量或矢量{1}
Uniform Random Number (Uniform RandomNumber)		
Minimum	最小值	标量或矢量{-1}
Maximum	最大值	标量或矢量{1}
Seed	初始种子	标量或矢量{0}
SampleTime	采样时间	标量或矢量{0}

附表 4 Sinks 库模块参数

模块(模块类型)/参数	对话框提示	值
Display (Display)		
Format	格式	{short} | long | short_e | long_e| bank
Decimation	抽取率(十分之……)	标量{1}
Floating	流显	{off} | on
SampleTime	采样时间	标量(采样周期){-1}或矢量[周期偏置]
Scope (Scope)		
Location	Scope 窗口在显示屏上的位置,大小	矢量{[left top right bottom]}
Open	若当模型打开后 Scope 展开。不能从对话框上设置	{off} | on
NumInputPorts	轴数	大于 0 的整数
TickLabels	隐藏点标识	{on} | off
ZoomMode	(初按缩放键)	{on} | xonly | yonly
AxesTitles	标题(右击鼠标)	标量{auto}
Grid	(后用)	{on} | off
TimeRange	时间范围	标量{auto}
YMin	Y 最小值	标量{-5}
YMax	Y 最大值	标量{5}
SaveToWorkspace	将数据存入工作空间	{off} | on
SaveName	变量名	变量{ScopeData}
DataFormat	格式	{matrix | structure}
LimitMaxRows	限定到目前的行数	{on} | off
MaxRows	(无标识)	标量{5000}
Decimation	(选中 Decimation 后的值)	标量{1}
SampleInput	(与 Decimation 转换)	{off} | on
SampleTime	(SampleInput 值)	标量(采样周期){0}或矢量[周期偏置]
Stop Simulation (StopSimulation)	(无模块特有参数)	
To File (ToFile)		
Filename	文件名	文件名{untitled.mat}
MatrixName	变量名	变量{ans}
Decimation	抽取率(十分之……)	标量{1}

续附表 4

模块(模块类型)/参数	对话框提示	值
SampleTime	采样时间	标量(采样周期){－1}或矢量[周期偏置]
To Workspace (ToWorkspace)		
VariableName	变量名	变量{simout}
Buffer	最大行数	标量{inf}
Decimation	抽取率(十分之……)	标量{1}
SampleTime	采样时间	标量(采样周期){－1}或矢量[周期偏置]
XY Graph(XY scope.)(封装)		

附表 5　Discrete 库模块参数

模块(模块类型)/参数	对话框提示	值
Discrete Filter(DiscreteFilter)		
Numerator	分子	矢量{[1]}
Denominator	分母	矢量{[1 2]}
SampleTime	采样时间	标量(采样周期){1}或矢量[周期偏置]
Discrete State-Space (DiscreteStateSpace)		
A	A	矩阵{1}
B	B	矩阵{1}
C	C	矩阵{1}
D	D	矩阵{1}
X0	初始条件	矢量{0}
SampleTime	采样时间	标量(采样周期){1}或矢量[周期偏置]
Discrete-Time Integrator (DiscreteIntegrator)		
IntegratorMethod	积分器算法	{ForwardEuler} | BackwardEuler | Trapezoidal
ExternalReset	外部设置	{none} | rising | falling | either
InitialConditionSource	初始条件源	{internal} | external
InitialCondition	初始条件	标量或矢量{0}
LimitOutput	输出极限	{off} | on
UpperSaturationLimit	上饱和极限	标量或矢量{inf}
LowerSaturationLimit	下饱和极限	标量或矢量{－inf}
ShowSaturationPort	显示饱和端口	{off} | on
ShowStatePort	显示状态端口	{off} | on
SampleTime	采样时间	标量(采样周期){1}或矢量[周期偏置]

续附表 5

模块(模块类型)/参数	对话框提示	值
Discrete Transfer Fcn (DiscreteTransfer-Fcn)		
Numerator	分子	矢量{[1]}
Denominator	分母	矢量{[1 0.5]}
SampleTime	采样时间	标量(采样周期) {1}或矢量[周期偏置]
Discrete Zero-Pole (DiscreteZeroPole)		
Zeros	零点	矢量{[1]}
Poles	极点	矢量{[0 0.5]}
Gain	增益	标量{1}
SampleTime	采样时间	标量(采样周期) {1}或矢量[周期偏置]
First-Order Hold (First Order Hold)(封装)		
Unit Delay (UnitDelay)		
X0	初始条件	标量或矢量{0}
SampleTime	采样时间	标量(采样周期) {1}或矢量[周期偏置]
Zero-Order Hold (ZeroOrderHold)		
SampleTime	采样时间	标量(采样周期) {1}或矢量[周期偏置]

附表 6 Continuous 库模块参数

模块(模块类型)/参数	对话框提示	值
Derivative (Derivative) (无模块特有参数)		
Integrator (Integrator)		
ExternalReset	外部设置	{none} | rising | falling | either
InitialConditionSource	初始条件源	{internal} | external
InitialConditionSource	初始条件源	{internal} | external
InitialCondition	初始条件	标量或矢量{0}
LimitOutput	输出极限	{off} | on
UpperSaturationLimit	上饱和极限	标量或矢量{inf}
LowerSaturationLimit	下饱和极限	标量或矢量{-inf}
ShowSaturationPort	显示饱和端口	{off} | on
ShowStatePort	显示状态端口	{off} | on
AbsoluteTolerance	绝对公差	标量或矢量{0}
Memory (Memory)		
X0	初始条件	标量或矢量{0}
InheritSampleTime	继承采样时间	{off} | on

续附表 6

模块(模块类型)/参数	对话框提示	值
Discrete State-Space (DiscreteStateSpace)		
A	A	矩阵{1}
B	B	矩阵{1}
C	C	矩阵{1}
D	D	矩阵{1}
X0	初始条件	矢量{0}
Transfer Fcn (TransferFcn)		
Numerator	分子	矢量或矩阵{[1]}
Denominator	分母	矢量或矩阵{[1 1]}
Transport Delay (TransportDelay)		
DelayTime	时间延迟	标量或矢量{1}
InitialInput	初始输入	标量或矢量{0}
BufferSize	初始缓冲器尺寸	标量{1024}
Variable Transport Delay (VariableTransportDelay)		
MaximumDelay	最大延迟	标量或矢量{10}
InitialInput	初始输入	标量或矢量{0}
MaximumPoints	缓冲器尺寸	标量{1024}
Zero-Pole (ZeroPole)		
Zeros	零点	矢量{[1]}
Poles	极点	矢量{[0 −1]}
Gain	增益	标量{1}

附表 7 Math 库模块参数

模块(模块类型)/参数	对话框提示	值
Abs (Abs) (无模块特有参数)		
Algebraic Constraint (Algebraic Constraint)(封装)		
Combinatorial Logic (CombinatorialLogic)		
TruthTable	真值表	矩阵{[0 0;0 1;0 1;1 0; 0 1;1 0;1 0;1 1]}
Complex to Magnitude-Angle		
Complex to Real-Imag		
Dot Product (Dot Product)(封装)		
Gain (Gain)		

续附表 7

<table>
<tr><th>模块(模块类型)/参数</th><th>对话框提示</th><th>值</th></tr>
<tr><td>Gain</td><td>增益</td><td>标量或矢量{1}</td></tr>
<tr><td>Logical Operator (Logic)</td><td></td><td></td></tr>
<tr><td>Operator</td><td>运算符</td><td>{AND}|OR|NAND|NOR|XOR|NOT</td></tr>
<tr><td>Inputs</td><td>输入数</td><td>标量[2]</td></tr>
<tr><td>Magnitude-Angle to Complex</td><td></td><td></td></tr>
<tr><td>Math Function (Math)</td><td></td><td></td></tr>
<tr><td>Operator</td><td>函数</td><td>{exp} | log | log10 | square | sqrt | pow | reciprocal | hypot | rem | mod</td></tr>
<tr><td>Matrix Gain (Matrix Gain) (封装)</td><td></td><td></td></tr>
<tr><td>MinMax (MinMax)</td><td></td><td></td></tr>
<tr><td>Function</td><td>函数</td><td>{min} | max</td></tr>
<tr><td>Inputs</td><td>输入端口数</td><td>标量[1]</td></tr>
<tr><td>Product (Product)</td><td></td><td></td></tr>
<tr><td>Inputs</td><td>输入数</td><td>标量[2]</td></tr>
<tr><td>Relational Operator (RelationalOperator)</td><td></td><td></td></tr>
<tr><td>Operator</td><td>运算符</td><td>== | != | < | {<=} | >= | ></td></tr>
<tr><td>Rounding Function (Rounding)</td><td></td><td></td></tr>
<tr><td>Operator</td><td>函数</td><td>{floor} | ceil | round | fix</td></tr>
<tr><td>Sign (Signum) 无模块特有参数)</td><td></td><td></td></tr>
<tr><td>Slider Gain (SliderGain)(封装)</td><td></td><td></td></tr>
<tr><td>Sum (Sum)</td><td></td><td></td></tr>
<tr><td>Inputs</td><td>符号表</td><td>矢量或符号表{++}</td></tr>
<tr><td>Trigonometric Function (Trigonometry)</td><td></td><td></td></tr>
<tr><td>Operator</td><td>函数</td><td>{sin} | cos | tan | asin | acos | atan | atan2 | sinh | cosh | tanh</td></tr>
</table>

附表 8 Functions and Tables 库模块参数

<table>
<tr><th>模块(模块类型)/参数</th><th>对话框提示</th><th>值</th></tr>
<tr><td>Fcn (Fcn)</td><td></td><td></td></tr>
<tr><td>Expr</td><td>表达式</td><td>表达式{sin(u(1)*exp(2.3*(-u(2))))}</td></tr>
<tr><td>Look-up Table (Lookup)</td><td></td><td></td></tr>
<tr><td>InputValues</td><td>输入值矢量</td><td>矢量{[-5:5]}</td></tr>
<tr><td>OutputValues</td><td>输出值矢量</td><td>矢量{tanh([-5:5])}</td></tr>
</table>

续附表 8

模块(模块类型)/参数	对话框提示	值
Look-Up Table (2-D) (Lookup Table (2-D))		
RowIndex	行	矢量
ColumnIndex	列	矢量
OutputValues	表	2-D 矩阵
MATLAB Fcn (MATLABFcn)		
MATLABFcn	MATLAB 函数	MATLAB 函数{sin}
OutputWidth	输出宽度	标量或矢量{−1}
S-Function (S-Function)		
FunctionName	S-function 名	名{system}
Parameters	S-function 参数	额外需要参数

附表 9　Nonlinear 库模块参数

模块(模块类型)/参数	对话框提示	值
Backlash (Backlash)		
BacklashWidth	Deadband 宽	标量或矢量{1}
InitialOutput	初始输出	标量或矢量{0}
Coulomb & Viscous Friction (Coulombic and Viscous Friction)(封装)		
Dead Zone (DeadZone)		
LowerValue	滞后区起点	标量或矢量{−0.5}
UpperValue	滞后区末点	标量或矢量{0.5}
Manual Switch		
Multiport Switch (MultiPortSwitch)		
Inputs	输入数	标量或矢量{3}
Quantizer (Quantizer)		
QuantizationInterval	量化区间	标量或矢量{0.5}
Rate Limiter (RateLimiter)		
RisingSlewLimit	上升速率	标量或矢量{1}
FallingSlewLimit	下降速率	标量或矢量{−1}
Relay (Relay)		
OnSwitchValue	接通点	标量或矢量{eps}
OffSwitchValue	断开点	标量或矢量{eps}
OnOutputValue	接通时的输出	标量或矢量{1}

续附表 9

模块(模块类型)/参数	对话框提示	值
OffOutputValue	断开时的输出	标量或矢量{0}
Saturation (Saturate)		
UpperLimit	上限	标量或矢量{0.5}
LowerLimit	下限	标量或矢量{-0.5}
S-Function (S-Function)		
FunctionName	S-function 名	名{system}
Parameters	S-function 参数	额外需要参数
Sign (Signum) 无模块特有参数)		
Switch (Switch)		
Threshold	阈值	标量或矢量{0}

附表 10 Signals & Systems 库模块参数

模块(模块类型)/参数	对话框提示	值
Bus Selector		
Configurable Subsystem (SubSystem)		
Choice	模块选定	字符串
LibraryName	库名	字符串
Data Store Memory (DataStoreMemory)		
DataStoreName	数据存储名	tag {A}
InitialValue	初始值	矢量{0}
Data Store Read (DataStoreRead)		
DataStoreName	数据存储名	tag {A}
SampleTime	采样时间	标量(采样周期){-1}或矢量[周期偏置]
Data Store Write (DataStoreWrite)		
DataStoreName	数据存储名	tag {A}
SampleTime	采样时间	标量(采样周期){-1}或矢量[周期偏置]
Data Type Conversion		
Demux (Demux)		
Outputs	输出数	标量或矢量{3}
Enable (EnablePort)		
StatesWhenEnabling	使能时状态	{held} | reset
ShowOutputPort	显示输出端口	{off} | on
From (From)		
GotoTag	转到标签	标签 {A}

续附表 10

模块(模块类型),参数	对话框提示	值
Goto (Goto)		
GotoTag	转到标签	标签 {A}
TagVisibility	标签可视性	{local} | scoped | global
Goto Tag Visibility (GotoTagVisibility)		
GotoTag	转到标签	标签 {A}
Ground (Ground)(无模块特有参数)		
Hit Crossing (HitCross)		
HitCrossingOffset	过零偏置	标量或矢量{0}
HitCrossingDirection	过零方向	rising | falling | {either}
ShowOutputPort	显示输出端口	{on} | off
IC (InitialCondition)		
Value	初始值	标量或矢量{1}
In (Inport)		
Port	端口数	标量{1}
PortWidth	端口宽	标量{−1}
SampleTime	采样时间	标量(采样周期){−1}或矢量[周期偏置]
Merge		
Model Info		
Mux (Mux)		
Inputs	输入数	标量或矢量{3}
Out (Outport)		
Port	端口数	标量{1}
OutputWhenDisabled	使能输出	{held} | reset
InitialOutput	初始输出	标量或矢量{0}
Probe (Probe)		
ProbeWidth	探测宽度	{on} | off
ProbeSampleTime	探测采样时间	{on} | off
ProbeCompexSignal	探测复信号	{on} | off
Subsystem (SubSystem)		
ShowPortLabels	Show/Hide Port Labels Format 菜单中一条	{on} | off
Terminator (Terminator)(无模块特有参数)		
Trigger (TriggerPort)		
TriggerType	触发类型	{rising} | falling | either | function-call
ShowOutputPort	显示输出端口	{off} | on
Width (Width)(无模块特有参数)		

D. 封装参数

在第 6 章已经对封装参数做了介绍，为查阅方便，本节仍然列出描述封装模块的参数。附表 11 所列的封装参数对应 Mask Editor 对话框的参数。

附表 11 封装参数

参 数	对话框参数	值
MaskType	封装类型	字符串
MaskDescription	模块描述	字符串
MaskHelp	模块帮助	字符串
MaskPrompts	提示(见后)	字符串组
MaskPromptString	提示(见后)	定界字符串
MaskStyles	控制类型(见后)	单元数组{Edit} ; Checkbox | Popup
MaskStyleString	控制类型(见后)	{Edit} Checkbox | Popup
MaskVariables	变量(见后)	字符串
MaskInitialization	初始化命令	MATLAB 命令
MaskDisplay	绘图命令	显示命令
MaskIconFrame	图标结构(可见 on，不可见 off)	{on} | off
MaskIconOpaque	图标透明度(不透明 on，透明 off)	{on} off
MaskIconRotate	图标旋转(旋转 on，固定 off)	on {off}
MaskIconUnits	绘坐标	Pixel | {Autoscale} | Normalized
MaskValues	模块参数值(见后)	字符串组
MaskValueString	模块参数值(见后)	定界字符串
MaskTunableValues	Tunable 参数属性	字符串组
MaskTunableValue string	Tunable 参数属性	定界字符串

当用户使用 Mask Editor 创建一个封装模块的对话框时，应提供以下信息：

① 提示。在 Prompt 栏内输入；

② 保存参数值的变量。在 Variable 栏内输入；

③ 所建栏的类型。通过选定一个 Control type 来指定；

④ 在栏中输入的值是计算还是以文字的形式存储。通过选定一个 Assignment 类型指定。

封装模块以下列方式存储从对话框上指定的值：

① 所有对话框参数的 Prompt 栏的值以字符串的形式存储在 MaskPromptString 参数

中，而且它们的各自值用“|”分隔，如下例所示：

"Slope:|Intercept:"

② 所有对话框参数 Variable 栏的值以字符串的形式存储在 MaskVariables 参数中，且各自的赋值用“;”分隔。序号指出与一个变量关联的是哪种提示。序号前的一个特殊字符表示赋值(Assignment)类型;@表示计算(Evaluate);& 表示文字(Literal)。

例如，"a=@1;b=&2;"表示将第一个参数栏输入的值赋予变量 a，且在赋值前在 MATLAB 下计算;将第二个参数栏输入的值赋予变量 b，且以文字形式存储，也就是说该值是从对话框上输入的字符串。

③ 所有对话框参数 Control type 栏的值以一个字符串存储在 MaskStyleString 参数中，且各自的赋值用“,”分隔。Popup strings 值出现在 popup 类型之后，如下例所示：

"edit,checkbox,popup(red|blue|green)"

④ 参数值以字符串的形式存储在 MaskValueString 封装参数内，且它们的各自值用“|”分隔。值的顺序与在对话框中参数的出现顺序一致。例如，下面的语句定义参数栏提示以及赋予这些参数的值。

MaskPromptString "Slope:|Intercept:"

MaskValueString "2|5"

E. 模型文件格式及示例

模型文件实际上就是一个 ASCII 结构的文件，它包含了描述模型的关键字——参数值对。文件按层次顺序描述模型成分。

编者提示:Simulink 3.0 版本的模型文件显示不出中文字符，请读者注意。Simulink 4.0 已做了改进。

1. 模型文件

模型文件的结构如下所示：

```
Model {
    <Model Parameter Name> <Model Parameter Value>...
    BlockDefaults {<Block Parameter Name> <Block Parameter Value>...}
    AnnotationDefaults {<Annotation Parameter Name> <Annotation Parameter alue>...}
    System {<System Parameter Name> <System Parameter Value>...
        Block {<Block Parameter Name> <Block Parameter Value>...}
        Line {<Line Parameter Name> <Line Parameter Value>...
            Branch {<Branch Parameter Name> <Branch Parameter Value>...}
        }
        Annotation {<Annotation Parameter Name> <Annotation Parameter Value>...}
    }
}
```

我们可以把模型文件看成是由描述不同模型成分的段组成的：

- Model 段。定义模型参数；
- BlockDefaults 段。包含模型中模块的默认设置；
- AnnotationDefaults 段。包含模型中注解的默认设置；
- System 段。包含描述模型中每个系统(包括最高级系统和每一个子系统)的参数。每个 System 段包含模块、连线以及注解说明。

所有模型和模块的参数在前面已经做过描述。

(1) Model 段

Model 段位于模型文件的顶部，定义模型级参数的值。这些参数包括模型名、用于最后一次修改模型的 Simulink 版本，以及仿真参数。

(2) BlockDefaults 段

BlockDefaults 段在仿真参数后出现，定义模型内模块参数的默认值。这些值可以被在 Block 段中定义的各自模块参数取代。

(3) AnnotationDefaults 段

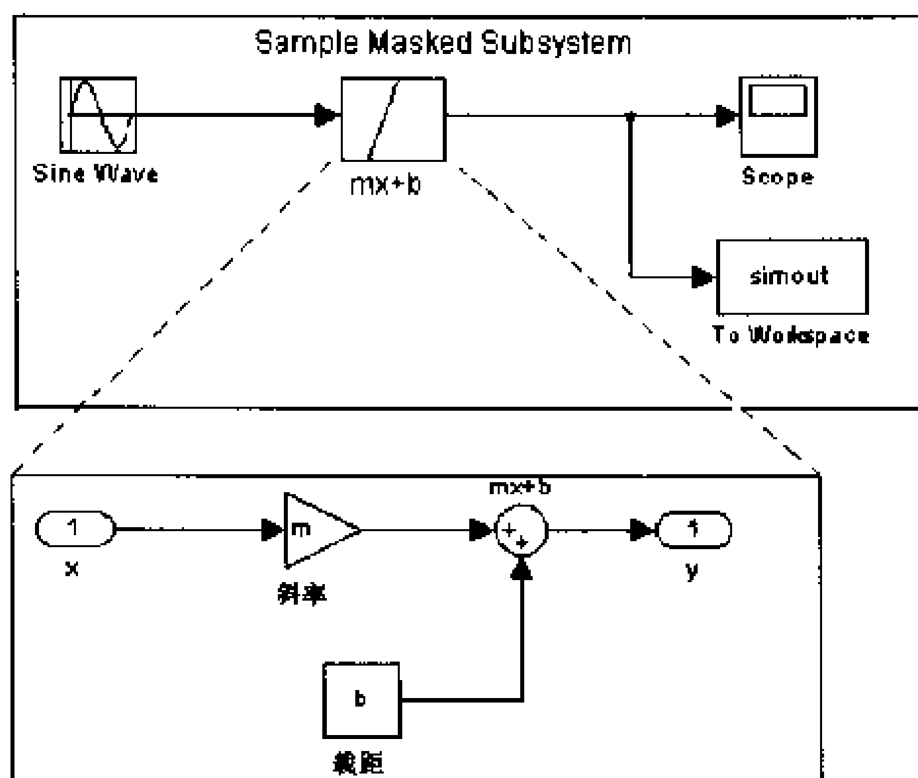
AnnotationDefaults 段出现在 BlockDefaults 段之后。本段定义模型中所有注解的默认参数。这些参数值不能使用 set_param 命令修改。

(4) System 段

模型中最高级系统和每个子系统各用一个 System 段描述。每个 System 段定义系统级参数并包括该系统中每个模块、连线及注解的 Block，Line 和 Annotation 段。每个包含一个分支点的 Line 包括一个定义分支连线的 Branch 段。

2. 一个采样模型文件

此模型文件描述一隐含采样系统(见第 6 章)。模型及其子系统如附图 1 所示。



附图 1　mx+b 模型

它的子系统文件名为 Example mask2. mdl。

模型文件为：Slopeinter4. mdl，现给出源码如下，仅供参考。

```
Model {
  Name                "slopeinter4"
  Version             3.00
  SimParamPage        "Solver"
  SampleTimeColors    off
  InvariantConstants  off
  WideVectorLines     off
  ShowLineWidths      off
  ShowPortDataTypes   off
  StartTime           "0.0"
  StopTime            "100.0"
  SolverMode          "Auto"
  Solver              "ode45"
  RelTol              "1e-3"
  AbsTol              "auto"
  Refine              "2"
  MaxStep             "auto"
  InitialStep         "auto"
  FixedStep           "auto"
  MaxOrder            5
  OutputOption        "RefineOutputTimes"
  OutputTimes         "[]"
  LoadExternalInput   off
  ExternalInput       "[t, u]"
  SaveTime            on
  TimeSaveName        "tout"
  SaveState           off
  StateSaveName       "xout"
  SaveOutput          on
  OutputSaveName      "yout"
  LoadInitialState    off
  InitialState        "xInitial"
  SaveFinalState      off
  FinalStateName      "xFinal"
  SaveFormat          "Matrix"
  LimitMaxRows        off
  MaxRows             "1000"
```

```
Decimation                "1"
AlgebraicLoopMsg          "warning"
MinStepSizeMsg            "warning"
UnconnectedInputMsg       "warning"
UnconnectedOutputMsg      "warning"
UnconnectedLineMsg        "warning"
InheritedTsInSrcMsg       "warning"
IntegerOverflowMsg        "warning"
UnnecessaryDatatypeConvMsg "none"
Int32ToFloatConvMsg       "warning"
SignalLabelMismatchMsg    "none"
ConsistencyChecking       "off"
ZeroCross                 on
SimulationMode            "normal"
BlockDataTips             on
BlockParametersDataTip    off
BlockAttributesDataTip    off
BlockPortWidthsDataTip    off
BlockDescriptionStringDataTip   off
BlockMaskParametersDataTip off
ToolBar                   on
StatusBar                 on
BrowserShowLibraryLinks off
BrowserLookUnderMasks     off
OptimizeBlockIOStorage    on
BufferReuse               on
BooleanDataType           off
RTWSystemTargetFile       "grt.tlc"
RTWInlineParameters       off
RTWRetainRTWFile          off
RTWTemplateMakefile       "grt_default_tmf"
RTWMakeCommand            "make_rtw"
RTWGenerateCodeOnly       off
ExtModeMexFile            "ext_comm"
ExtModeBatchMode          off
ExtModeTrigType           "manual"
ExtModeTrigMode           "normal"
ExtModeTrigPort           "1"
ExtModeTrigElement        "any"
```

```
ExtModeTrigDuration          1000
ExtModeTrigHoldOff          0
ExtModeTrigDelay          0
ExtModeTrigDirection          "rising"
ExtModeTrigLevel          0
ExtModeArchiveMode          "off"
ExtModeAutoIncOneShot          off
ExtModeIncDirWhenArm          off
ExtModeAddSuffixToVar          off
ExtModeWriteAllDataToWs off
ExtModeArmWhenConnect          on
Created          "Wed Feb 12 22:07:38 2003"
UpdateHistory          "UpdateHistoryNever"
ModifiedByFormat          "%<Auto>"
ModifiedDateFormat          "%<Auto>"
LastModifiedDate          "Wed Feb 12 22:58:15 2003"
ModelVersionFormat          "1. %<AutoIncrement:5>"
ConfigurationManager          "None"
BlockDefaults {
   Orientation          "right"
   ForegroundColor          "black"
   BackgroundColor          "white"
   DropShadow          off
   NamePlacement          "normal"
   FontName          "Helvetica"
   FontSize          10
   FontWeight          "normal"
   FontAngle          "normal"
   ShowName          on
}
AnnotationDefaults {
   HorizontalAlignment          "center"
   VerticalAlignment          "middle"
   ForegroundColor          "black"
   BackgroundColor          "white"
   DropShadow          off
   FontName          "Helvetica"
   FontSize          10
   FontWeight          "normal"
```

```
    FontAngle              "normal"
  }
  LineDefaults {
    FontName               "Helvetica"
    FontSize               9
    FontWeight             "normal"
    FontAngle              "normal"
  }
  System {
    Name                   "slopeinter4"
    Location               [81, 127, 581, 428]
    Open                   on
    ModelBrowserVisibility off
    ModelBrowserWidth      200
    ScreenColor            "automatic"
    PaperOrientation       "landscape"
    PaperPositionMode      "auto"
    PaperType              "A4"
    PaperUnits             "centimeters"
    ZoomFactor             "100"
    AutoZoom               on
    ReportName             "simulink-default.rpt"
    Block {
      BlockType            Scope
      Name                 "Scope"
      Ports                [1, 0, 0, 0, 0]
      Position             [370, 49, 400, 81]
      Floating             off
      Location             [188, 365, 512, 604]
      Open                 off
      NumInputPorts        "1"
      TickLabels           "OneTimeTick"
      ZoomMode             "on"
      List {
    ListType               AxesTitles
    axes1                  "%<SignalLabel>"
      }
      Grid                 "on"
      TimeRange            "auto"
```

```
    YMin              "-5"
    YMax              "5"
    SaveToWorkspace   off
    SaveName          "ScopeData"
    DataFormat        "StructureWithTime"
    LimitMaxRows      on
    MaxRows           "5000"
    Decimation        "1"
    SampleInput       off
    SampleTime        "0"
  }
  Block {
    BlockType         Sin
    Name              "Sine Wave"
    Position          [60, 50, 90, 80]
    Amplitude         "1"
    Frequency         "1"
    Phase             "0"
    SampleTime        "0"
  }
  Block {
    BlockType         ToWorkspace
    Name              "To Workspace"
    Position          [355, 110, 415, 140]
    VariableName      "simout"
    Buffer            "1000"
    Decimation        "1"
    SampleTime        "-1"
    SaveFormat        "Structure"
  }
  Block {
    BlockType         SubSystem
    Name              "mx+b"
    Ports             [1, 1, 0, 0, 0]
    Position          [175, 46, 220, 84]
    ShowPortLabels    on
    MaskType          "封装模块"
    MaskDescription   "模拟方程 y=mx+b"
    MaskHelp          "输入斜率(m)和截距(b)的值。系统将自动运算并"
```

```
                         "产生输出 y=mx+b。"
  MaskPromptString          "截距|斜率"
  MaskStyleString          "edit,edit"
  MaskTunableValueString   "on,on"
  MaskCallbackString       "|"
  MaskEnableString          "on,on"
  MaskVisibilityString     "on,on"
  MaskVariables            "b=@1;m=@2;"
  MaskDisplay              "plot([0 1],[0 m]+(m<0))"
  MaskIconFrame            on
  MaskIconOpaque           off
  MaskIconRotate           "none"
  MaskIconUnits            "normalized"
  MaskValueString          "3|2"
  System {
Name                "mx+b"
Location           [377, 74, 787, 427]
Open               off
ModelBrowserVisibility    off
ModelBrowserWidth      200
ScreenColor          "white"
PaperOrientation      "landscape"
PaperPositionMode      "auto"
PaperType           "A4"
PaperUnits           "centimeters"
ZoomFactor           "100"
AutoZoom            on
Block {
  BlockType              Inport
  Name                   "x"
  Position             [65, 103, 95, 117]
  Port                  "1"
  PortWidth              "-1"
  SampleTime              "-1"
  DataType              "auto"
  SignalType             "auto"
  Interpolate            on
}
Block {
```

```
    BlockType           Gain
    Name                "Slope"
    Position            [125, 95, 155, 125]
    Gain                "m"
    SaturateOnIntegerOverflow on
  }
  Block {
    BlockType           Sum
    Name                "Sum"
    Ports               [2, 1, 0, 0, 0]
    Position            [225, 100, 245, 120]
    ShowName            off
    IconShape           "round"
    Inputs              "|++"
    SaturateOnIntegerOverflow on
  }
  Block {
    BlockType           Constant
    Name                "intercept"
    Position            [160, 160, 190, 190]
    Value               "b"
  }
  Block {
    BlockType           Outport
    Name                "y"
    Position            [315, 103, 345, 117]
    Port                "1"
    OutputWhenDisabled  "held"
    InitialOutput       "[]"
  }
  Line {
    SrcBlock            "x"
    SrcPort             1
    DstBlock            "Slope"
    DstPort             1
  }
  Line {
    SrcBlock            "intercept"
    SrcPort             1
```

```
    Points          [40, 0]
    DstBlock         "Sum"
    DstPort         2
  }
  Line {
    SrcBlock         "Slope"
    SrcPort         1
    DstBlock         "Sum"
    DstPort         1
  }
  Line {
    SrcBlock         "Sum"
    SrcPort         1
    DstBlock         "y"
    DstPort         1
  }
    }
  }
  Line {
    SrcBlock              "Sine Wave"
    SrcPort              1
    DstBlock              "mx+b"
    DstPort              1
  }
  Line {
    SrcBlock              "mx+b"
    SrcPort              1
    Points              [80, 0]
    Branch {
  DstBlock         "Scope"
  DstPort           1
    }
    Branch {
  Points          [0, 60]
  DstBlock        "To Workspace"
  DstPort           1
    }
  }
  Annotation {
```

```
        Position          [217, 25]
        Text              "Sample Masked Subsystem"
      }
    }
  }
```

参考文献

1 . The MathWorks,Inc. Using Simulink(Simulink 3). January 1999

2 The MathWorks,Inc. Using Simulink(Simulink 4). November 2000

3 The MathWorks,Inc. Using Simulink(Simulink 4.1). June 2001

4 施阳,严卫生,李俊等. MATLAB 语言精要及动态仿真工具 SIMULINK. 西安:西北工业大学出版社,1997

［Ｇｅｎｅｒａｌ Ｉｎｆｏｒｍａｔｉｏｎ］
书名＝基于ＭＡＴＬＡＢ的动态模型与系统仿真工具——Ｓｉｍｕｌｉｎｋ ３．０／４．Ｘ
作者＝
页数＝３２８
ＳＳ号＝１１２１８２６７
出版日期＝

封面
书名
版权
前言
目录
正文